

Architecture des réseaux



- L'exposé des principales caractéristiques des réseaux : protocoles, architectures de communication, réseaux locaux, TCP/IP
- Une centaine d'exercices avec leurs corrigés détaillés
- Sur www.pearson.fr : des exercices complémentaires et un chapitre sur la sécurité des réseaux

collection Synthex



Danièle DROMARD
Dominique SERET

Informatique

Synthèse de cours



exercices corrigés

Architecture des réseaux

Danièle Dromard

Université Pierre et Marie Curie (Paris 6)

et Dominique Seret

Université René Descartes (Paris 5)

collection Synthex



ISBN : 978-2-7440-7385-4
ISSN : 1768-7616

Composition sous FrameMaker : IDT

Sommaire

Les auteurs

Danièle DROMARD, maître de conférences à l'université Pierre et Marie-Curie (Paris 6). Son domaine d'enseignement et de recherche concerne les architectures informatiques et les réseaux. Elle est responsable de l'unité d'enseignement « introduction aux réseaux » en troisième année de licence d'informatique. En outre, elle enseigne les principes de base de l'architecture des ordinateurs dans l'unité d'enseignement « machines et représentation ». Elle a publié plusieurs ouvrages sur les réseaux informatiques, dont *Réseaux et télématique*, Eyrolles, *Réseaux informatiques, cours et exercices*, Eyrolles, *L'Architecture SNA*, Eyrolles.

Dominique SERET, professeur à l'université René-Descartes (Paris 5), est directrice de l'Unité de Formation et de Recherche en mathématiques et informatique. Elle enseigne la logique, l'algorithmique et l'introduction aux réseaux en licence d'informatique, ainsi que la sécurité des réseaux en master MIAGE. Son domaine de recherche concerne plus particulièrement les réseaux et l'évaluation des performances. Elle a publié plusieurs ouvrages en informatique, dont *Réseaux et télématique*, Eyrolles, *Réseaux informatiques, cours et exercices*, Eyrolles, *RNIS, description technique*, Masson, *Introduction aux réseaux*, Hermès.

Ensemble, elles ont écrit plusieurs articles pour l'*Encyclopaedia Universalis*.

Introduction

Les réseaux informatiques sont devenus incontournables aujourd'hui. Ils sont employés dans toutes les entreprises et même chez les particuliers. Ils permettent de mettre en œuvre des applications très diverses, des plus simples aux plus sophistiquées. La plus connue est la navigation sur le Web, c'est-à-dire le partage d'informations grâce à Internet.

Qu'il s'agisse de réseaux locaux, de réseaux sans fil, de réseaux d'opérateurs ou de petits réseaux privés, ils obéissent tous à des principes de structuration qu'il est indispensable de comprendre. Ils utilisent une architecture en couches, dans laquelle la communication entre ordinateurs obéit à des règles précises définies par des protocoles de communication. Les protocoles les plus connus sont TCP et IP, ils ont donné leur nom à l'architecture TCP/IP.

Le *Synthex Architecture des réseaux* offre un cadre pratique qui permet d'acquérir la maîtrise des réseaux informatiques, en associant étroitement l'étude des mécanismes de communication à celle des protocoles. Après avoir présenté les supports de transmission et le codage des signaux en ligne, il étudie chacune des couches de protocoles en proposant des exercices adaptés à chaque notion. L'ouvrage expose les fondamentaux des architectures de réseaux et présente les notions d'adressage, de routage et d'interconnexion de réseaux. Cette présentation est accompagnée de nombreux exemples et exercices qui montrent la puissance du principe de structuration en couches et de l'encapsulation.

Cet ouvrage est issu d'un enseignement dispensé de nombreuses fois à des étudiants en formation initiale mais aussi des apprenants en formation continue. Il a l'ambition de répondre à l'attente de tous ceux qui veulent comprendre le fonctionnement des réseaux et de leurs protocoles.

Le plan

Les architectures de réseaux informatiques et leurs protocoles sont exposés au cours de neuf chapitres de la façon suivante :

Chapitre 1 : **Transmissions et supports.** Ce chapitre présente les éléments de base de la transmission et montre comment les signaux électriques, lumineux ou électromagnétiques, se propagent dans des supports comme les câbles ou les fibres optiques et permettent ainsi la communication entre équipements informatiques à distance les uns des autres.

Chapitre 2 : **Les protocoles de liaison de données.** Centré sur les mécanismes de base de la communication entre deux équipements informatiques, le contrôle de la validité des messages transmis et du rythme de la transmission, l'utilisation de la temporisation, ce chapitre montre le rôle du protocole de liaison de données.

Chapitre 3 : **Les concepts généraux des réseaux.** Il généralise la communication à plusieurs équipements pour constituer un réseau. Il expose les besoins d'adressage, de routage et de partage des ressources entre les différentes communications. Il détaille les différentes solutions de commutation mises en place pour plusieurs exemples de réseaux : réseaux téléphoniques, réseaux de données, Internet.

Chapitre 4 : **Les architectures de communication.** Il montre l'intérêt de la normalisation pour la définition d'une architecture en couches et aborde les variantes conçues pour les réseaux locaux et Internet.

Chapitre 5 : **Les réseaux locaux d'entreprise.** Présents partout, ils constituent l'environnement initial de toutes les entreprises et de tous les particuliers pour accéder à Internet. Ethernet est le produit le plus répandu. Ce chapitre détaille son fonctionnement ainsi que ses évolutions vers des débits plus élevés et vers l'utilisation des commutateurs. Il explique également les spécificités des réseaux sans fils.

Chapitre 6 : **Le protocole IP.** C'est le protocole phare de l'architecture TCP/IP. Ce chapitre explique son fonctionnement mais aussi ses limitations. Il montre comment un datagramme est traité dans l'interconnexion de réseaux que constitue Internet.

Chapitre 7 : **Les protocoles de transport.** Pour l'utilisateur, la qualité du service rendu par Internet peut être insuffisante. Ce chapitre montre comment un protocole de transport comme TCP pallie les défaillances du réseau. Il illustre la récupération des messages perdus, la détection des messages dupliqués et le contrôle de flux ou de débit.

Chapitre 8 : **Le routage.** Ce chapitre montre les problèmes spécifiques de recherche d'un chemin à travers un réseau et explique comment les routeurs communiquent entre eux pour partager les informations sur l'état des liaisons du réseau. Il illustre par des exemples les deux principaux algorithmes de recherche du plus court chemin.

Chapitre 9 : **Les applications.** Ce chapitre décrit les principales applications qui ont justifié la construction des architectures de communication : le courrier électronique, le transfert de fichiers, la navigation sur le Web.

Le lecteur pourra également trouver sur le site www.pearsoneducation.fr deux chapitres supplémentaires :

Sécurité et mobilité. Ce chapitre aborde les différents services de sécurité et les mécanismes mis en place pour assurer cette sécurité : le chiffrement, les signatures numériques, les certificats, les pare-feu…

Études de cas. Ce chapitre aborde des aspects transversaux, avec une approche en couches conforme à l'architecture des réseaux.

Les exercices, qui occupent la moitié du livre, sont intégralement corrigés et permettent au lecteur d'appréhender, de façon progressive, toutes les notions de base des architectures de réseaux. Tous les énoncés sont le fruit d'une expérience pédagogique diversifiée. Ils ont été testés et ont prouvé leur efficacité.



Les transmissions et les supports

Un réseau suppose plusieurs équipements informatiques (ordinateurs…) situés à distance les uns des autres. La première chose à mettre en œuvre pour constituer le réseau est la transmission des informations d'un équipement à l'autre : on utilise, pour cela, des supports de transmission dont nous présentons les caractéristiques principales dans les deux premières sections de ce chapitre. De plus, à chaque nature de support correspond une forme particulière du signal qui s'y propage. Il faut donc fabriquer les signaux, grâce à l'équipement communément appelé « modem ». Les techniques de transmission et l'interface entre l'ordinateur et son modem sont normalisées pour assurer l'interopérabilité des équipements. À titre d'exemple, nous décrivons brièvement le raccordement ADSL dans la dernière section.

1 Supports de transmission

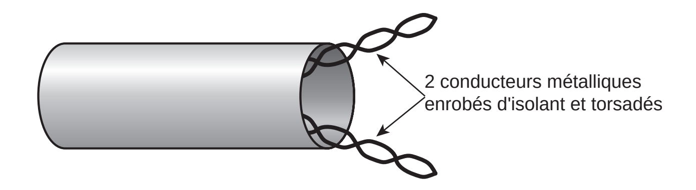
Les supports de transmission (décrits brièvement dans cette première section) sont nombreux. Parmi ceux-ci, trois familles sont à distinguer : les supports métalliques, non métalliques et immatériels. Les supports métalliques, comme les paires torsadées et les câbles coaxiaux, sont les plus anciens, les plus largement utilisés et servent à transmettre des courants électriques. Les supports de verre ou de plastique, comme les fibres optiques, transmettent de la lumière, tandis que les supports immatériels des *communications sans fil* transmettent des ondes électromagnétiques et sont en plein essor.

1.1 Paires torsadées

Une *paire torsadée non blindée* (UTP, *Unshielded Twisted Pair*) se compose de deux conducteurs en cuivre, isolés l'un de l'autre et enroulés de façon hélicoïdale autour de l'axe de symétrie longitudinal (voir figure 1.1).

Figure 1.1

Paire torsadée.



L'enroulement réduit les conséquences des inductions électromagnétiques parasites provenant de l'environnement. L'utilisation la plus courante de la paire torsadée est le raccordement des usagers au central téléphonique (la *boucle locale*) ou la desserte des usagers de réseaux privés. Son principal inconvénient est l'affaiblissement des courants transmis, d'autant plus important que le diamètre des conducteurs est faible. Les paires torsadées contiennent, à intervalles réguliers, des éléments appelés *répéteurs* qui régénèrent les signaux transmis. Quand plusieurs paires sont rassemblées dans un même câble, les courants qu'elles transportent interfèrent les uns avec les autres. Ce phénomène est appelé *diaphonie*.

Pour les réseaux locaux d'entreprise, où les distances se limitent à quelques kilomètres, la paire torsadée peut suffire. Ses avantages sont nombreux : technique maîtrisée, facilité de connexion et d'ajout de nouveaux équipements, faible coût. Certains constructeurs proposent des *paires torsadées blindées* (STP, *Shielded Twisted Pair*). Enrobées d'un conducteur cylindrique, elles sont mieux protégées des rayonnements électromagnétiques parasites. Une meilleure protection prévoit un blindage par paire.

1.2 Câbles coaxiaux

Pour éviter les perturbations dues aux bruits externes, on utilise souvent deux conducteurs métalliques cylindriques de même axe séparés par un isolant. Le tout forme un ensemble appelé *câble coaxial* (voir figure 1.2). Ce câble présente de meilleures performances que la paire torsadée : affaiblissement moindre, transmission de signaux de fréquences plus élevées, etc.

La capacité de transmission d'un câble coaxial dépend de sa longueur et des caractéristiques physiques des conducteurs et de l'isolant. Sur 1 km, un débit de plusieurs dizaines de Mbit/s peut être atteint alors que sur des distances plus courtes, des débits supérieurs sont possibles. Sur des distances supérieures à 10 km, les débits de transmission sont inférieurs à 10 kbit/s.



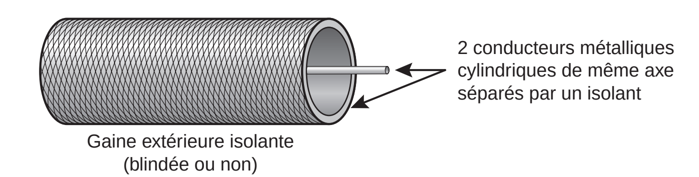


Figure 1.2

Câble coaxial.

1.3 Fibre optique

Une *fibre optique* est constituée d'un fil de verre très fin. Elle comprend un cœur, dans lequel se propage la lumière émise par une diode électroluminescente ou une source laser (voir figure 1.3), et une gaine optique dont l'indice de réfraction garantit que le signal lumineux reste dans la fibre.

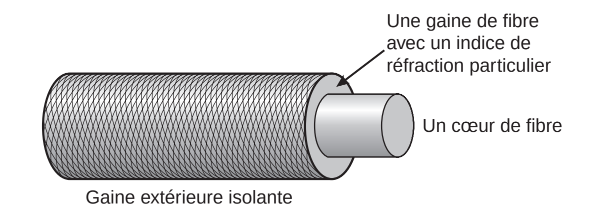


Figure 1.3

Fibre optique.

Les avantages de la fibre optique sont nombreux : le diamètre extérieur est de l'ordre de 0,1 mm, son poids de quelques grammes au kilomètre. Cette réduction de taille et de poids la rend facilement utilisable. En outre, sa très grande capacité permet la transmission simultanée de très nombreux canaux de télévision, de téléphone… Les points de régénération des signaux transmis sont plus éloignés, du fait de l'atténuation plus faible de la lumière. Enfin, l'insensibilité des fibres aux parasites électromagnétiques constitue un avantage très apprécié, puisqu'une fibre optique supporte sans difficulté la proximité d'émetteurs radioélectriques. On peut donc les utiliser dans des environnements très perturbés (avec de puissants champs électromagnétiques, par exemple) ou pour isoler électriquement des bâtiments entre eux.

1.4 Transmissions sans fil

Les ondes électromagnétiques se propagent dans l'atmosphère ou dans le vide (le terme d'*éther* désigne parfois ce type de support). L'absence de support matériel apporte une certaine souplesse et convient aux applications comme la téléphonie ou les télécommunications mobiles, sans nécessiter la pose coûteuse de câbles. On utilise des faisceaux directifs, faisceaux hertziens (pour franchir de grandes distances) ou ondes diffusées (pour atteindre des récepteurs géographiquement dispersés).

Faisceaux hertziens

Les *faisceaux hertziens* reposent sur l'utilisation de fréquences très élevées (de 2 GHz à 15 GHz et jusqu'à 40 GHz) et de faisceaux directifs produits par des antennes directionnelles qui émettent dans une direction donnée. La propagation des ondes est limitée à l'horizon optique ; la transmission se fait entre des stations placées en hauteur, par exemple sur une tour ou au sommet d'une colline, pour éviter les obstacles dus aux constructions environnantes. Les faisceaux hertziens s'utilisent pour la transmission par satellite, pour celle des chaînes de télévision ou pour constituer des artères de transmission longue distance dans les réseaux téléphoniques.

Ondes radioélectriques

Les *ondes radioélectriques* correspondent à des fréquences comprises entre 10 kHz et 2 GHz. Un émetteur diffuse ces ondes captées par des récepteurs dispersés géographiquement. Contrairement aux faisceaux hertziens, il n'est pas nécessaire d'avoir une visibilité directe entre émetteur et récepteur, car celui-ci utilise l'ensemble des ondes réfléchies et diffractées. En revanche, la qualité de la transmission est moindre car les interférences sont nombreuses et la puissance d'émission beaucoup plus faible.

Remarque

L'attribution des bandes de fréquences varie selon les continents et fait l'objet d'accords internationaux. Le tableau 1.1 donne les grandes lignes de la répartition des ondes en France. On constate que le découpage est complexe et qu'il reste peu de place pour de nouvelles applications.

Tableau 1.1

Affectation des fréquences en France

Gamme de fréquences	Type d'utilisation
10 kHz – 150 kHz	Communications radiotélégraphiques
150 kHz – 300 kHz	Radiodiffusion (grandes ondes)
510 kHz – 1605 kHz	Radiodiffusion (petites ondes)
6 MHz – 20 MHz	Radiodiffusion (ondes courtes)
29,7 MHz – 41 MHz	Radiotéléphonie
47 MHz – 68 MHz	Télévision
68 MHz – 87,5 MHz	Liaisons radio en modulation de fréquence
87,5 MHz – 108 MHz	Radiodiffusion
108 MHz – 162 MHz	Radiotéléphonie
162 MHz – 216 MHz	Télévision
216 MHz – 470 MHz	Radiotéléphonie
470 MHz – 860 MHz	Télévision et radar
860 MHz – 960 MHz	Radiotéléphonie
Autour de 1 800 MHz	Radiotéléphonie
Entre 6 et 30 GHz	Services satellites en fixe

2 Caractéristiques globales des supports de transmission

Quelle que soit la nature du support, le terme *signal* désigne le courant, la lumière ou l'onde électromagnétique transmis. Certaines caractéristiques physiques des supports en perturbent la transmission. La connaissance de leurs caractéristiques (la bande passante, la sensibilité aux bruits, les limites des débits possibles) est donc nécessaire pour fabriquer de « bons » signaux, c'est-à-dire les mieux adaptés aux supports utilisés.

2.1 Bande passante

Les supports ont une *bande passante* limitée. Certains signaux s'y propagent correctement (ils sont affaiblis mais reconnaissables à l'autre extrémité), alors que d'autres ne les traversent



pas (ils sont tellement affaiblis ou déformés qu'on ne les reconnaît plus du tout à la sortie). Intuitivement, plus un support a une bande passante large, plus il transporte d'informations par unité de temps.

Définition

La bande passante est la bande de fréquences dans laquelle les signaux appliqués à l'entrée du support ont une puissance de sortie supérieure à un seuil donné (après traversée du support de transmission). Le seuil fixé correspond à un rapport déterminé entre la puissance du signal d'entrée et la puissance du signal trouvé à la sortie (voir figure 1.4).

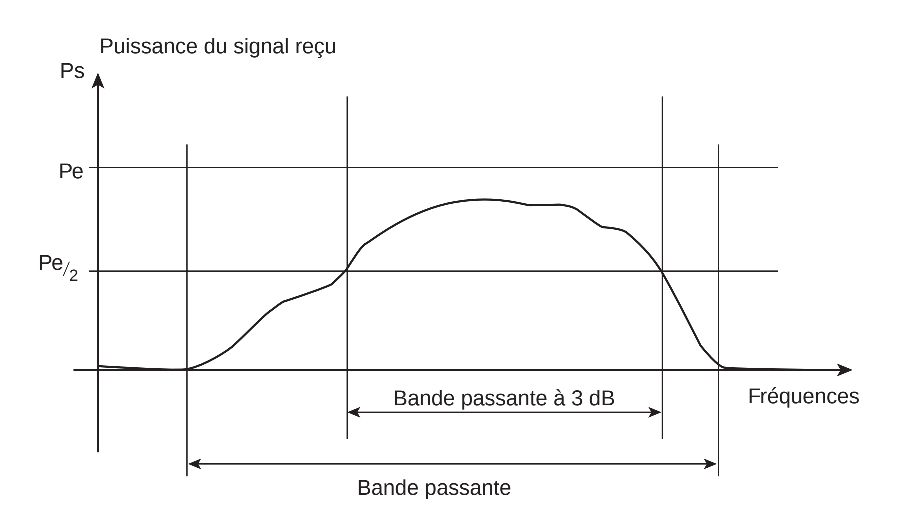


Figure 1.4
Notion de bande passante.

Remarque

En général, on caractérise un support par sa bande passante à 3 dB (décibels), c'est-à-dire par la plage de fréquences à l'intérieur de laquelle la puissance de sortie est, au pire, divisée par deux. En notant P_s la puissance de sortie, et P_e la puissance d'entrée, l'affaiblissement A en dB est donné par la formule :

$A = 10 * \log_{10} P_e/P_s$. Pour $P_e/P_s = 2$, on trouve : $10 * \log_{10} P_e/P_s = 3$ dB.

2.2 Bruits et distorsions

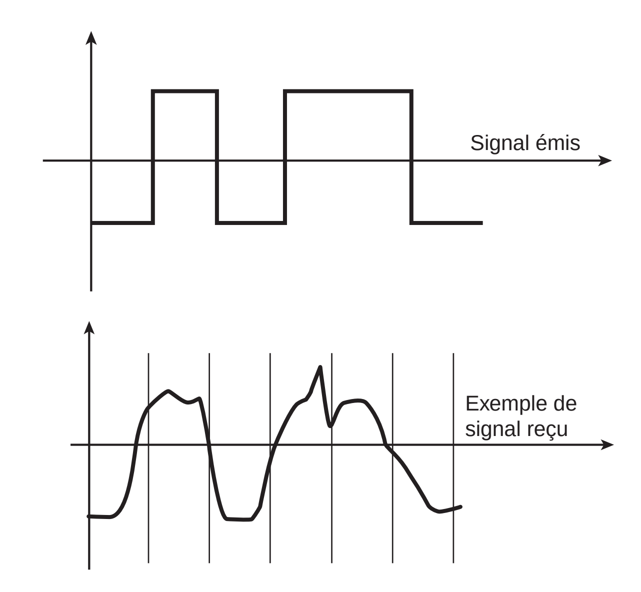
Les supports de transmission déforment les signaux qu'ils transportent, même lorsqu'ils ont des fréquences adaptées, comme le montre la figure 1.5. Diverses sources de *bruit* peuvent perturber les signaux : parasites, phénomènes de diaphonie… Certaines perturbations de l'environnement introduisent également des bruits (foudre, orages pour le milieu aérien, champs électromagnétiques dans des ateliers pour les supports métalliques…).

Par ailleurs, les supports affaiblissent et retardent les signaux. Par exemple, la distance est un facteur d'affaiblissement, particulièrement important pour les liaisons par satellite. Ces déformations, appelées *distorsions*, peuvent être gênantes pour la bonne reconnaissance des signaux en sortie, d'autant qu'elles varient généralement avec la fréquence des signaux émis.

Même lorsque les signaux sont adaptés aux supports de transmission, on ne peut pas garantir leur réception correcte à 100 %. Le récepteur d'un signal doit prendre une décision dans un laps de temps très court. De ce fait, cette décision peut être mauvaise.

Figure 1.5

Signal émis et exemple de signal reçu.



Par exemple, un symbole 1 émis peut donner une décision « symbole 0 reçu », ce qui constitue une *erreur de transmission*. Les fibres optiques sont les meilleurs supports, car le taux d'erreur y est très faible : 10^{-12} (c'est-à-dire une mauvaise décision pour 10^{12} bits transmis). Les câbles et les supports métalliques présentent des taux d'erreur moyens. Les liaisons sans fil ont un taux d'erreur variable, sensible aux conditions météorologiques.

2.3 Capacité limitée des supports de transmission

La *capacité* d'un support de transmission mesure la quantité d'informations transportée par unité de temps. L'ensemble des caractéristiques que nous venons de voir fait que la capacité d'un support est limitée. Un théorème dû à Shannon[1] exprime, en bits par seconde, la borne maximale de la capacité Cap_{Max} d'un support de transmission :

$$Cap_{Max} = W{*}\log_2 (1 + S/B).$$

Dans cette formule, *W* est la largeur de la bande passante du support de transmission exprimée en hertz, *S*/*B* représente la valeur du rapport entre la puissance du signal (notée *S*) et la puissance du bruit (notée *B*) ; la base 2 du logarithme sert à exprimer la quantité d'informations en bits (voir section 4.2).

Exemple

Sur une liaison téléphonique dont la bande passante a une largeur de 3 100 Hz et un rapport S/B correspondant à 32 dB (valeurs courantes), on obtient :
$10 \log_{10} S/B = 32$ donc $\log_{10} S/B = 3{,}2$ soit $S/B = 1585$;
$Cap_{Max} = 3100{*}\log_2 (1 + 1585)$;
comme $1586 = 2^{10{,}63}$, $Cap_{Max} = 3100{*}10{,}63 = 33\ 000$ bit/s.

Le choix d'un support de transmission tient compte de nombreux éléments. Des considérations économiques (le prix de revient du support, le coût de sa maintenance, etc.) interviennent en plus des facteurs techniques que nous venons de présenter, de même que la nature des signaux propagés, puisque l'équipement de transmission de données contient une partie spécifique au support de transmission utilisé. Examinons maintenant les techniques de transmission du signal véhiculant les données sur le support.

1. Claude Shannon (1916-2001), mathématicien américain qui a développé la théorie de l'information.



3 Fabrication des signaux : techniques de transmission

Selon les techniques de transmission utilisées, un équipement spécifique est placé à chaque extrémité du support : soit un *modem* (modulateur-démodulateur), soit un *codec* (codeur-décodeur). Cet équipement assure la fabrication des signaux en émission et leur récupération en réception. Pour émettre les données, le modem reçoit la suite de données binaires à transmettre et fournit un signal dont les caractéristiques sont adaptées au support de transmission. Inversement, en réception, le modem extrait la suite des données binaires du signal reçu. Le support de transmission est ainsi transparent à l'utilisateur. Le support de transmission et les deux modems placés à chacune de ses extrémités constituent un ensemble appelé *circuit de données*, comme le montre la figure 1.6.

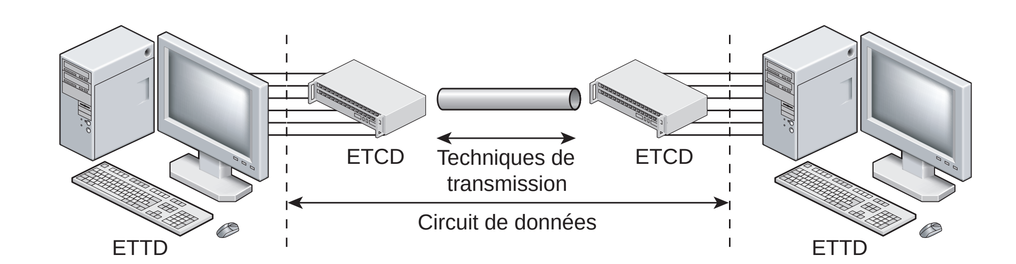


Figure 1.6

Équipements constitutifs d'un circuit de données.

L'ISO[2] et l'ITU (*International Telecommunications Union*) ont attribué des appellations génériques normalisées au modem et à l'équipement qui émet ou reçoit les données (ordinateur de l'utilisateur, imprimante…). Ainsi, le modem et le codec appartiennent à la famille des ETCD (équipement de terminaison du circuit de données), l'ordinateur ou l'imprimante font partie des ETTD (équipement terminal de traitement des données).

Définition

Le *circuit de données* est une entité capable d'envoyer ou de recevoir une suite de données binaires, à un débit donné, dans un délai donné et avec un taux d'erreur dépendant du support utilisé.

L'ETTD émetteur fournit à l'ETCD, régulièrement dans le temps, les données à transmettre. L'ETCD les émet sous forme d'un signal à deux valeurs (correspondant à 0 et 1), appelé *message de données synchrone* (voir figure 1.7). En effet, les intervalles de temps alloués à chaque symbole sont égaux et coïncident avec les périodes successives d'une base de temps (ou horloge) indispensable à l'interprétation du message de données.

2. Le nom de l'organisation donnerait lieu à des abréviations différentes selon les langues (IOS pour *International Organisation for Standardization* en anglais, OIN pour *Organisation internationale de normalisation* en français…). Il a été décidé d'emblée d'adopter un mot provenant du grec *isos* (égal), pour que la forme abrégée du nom de l'organisation soit toujours ISO.

Remarque

L'utilisation d'un circuit de données dépend de la nature des ETCD situés aux extrémités du support de transmission. La communication est en mode *duplex intégral* si la transmission simultanée est possible dans les deux sens. Si elle n'est possible que dans un seul sens à un moment donné (transmission *à l'alternat*), le circuit est *semi-duplex*. Enfin, le circuit est *simplex* lorsque la transmission ne se fait que dans un seul sens prédéfini.

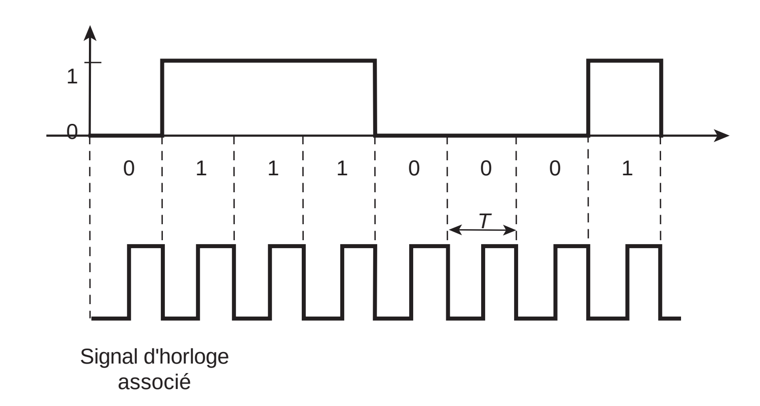


Figure 1.7

Message de données synchrone.

Le message de données synchrone utilise une représentation conventionnelle de l'information. La plus habituelle est un signal binaire sans retour à zéro, dit NRZ (*No Return to Zero*). On utilise un niveau de tension (+*a*, figure 1.8) pendant une période complète pour représenter la valeur 1 d'un bit, et un autre niveau (–*a*, figure 1.8) pour sa valeur 0.

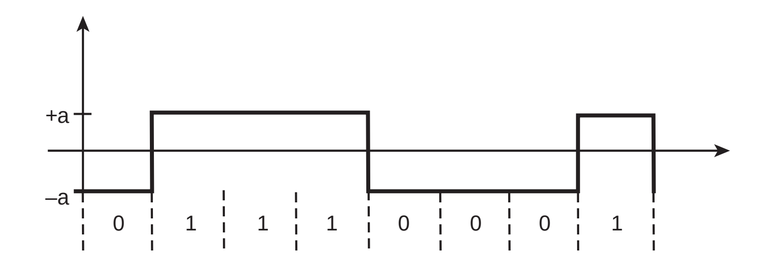


Figure 1.8

Représentation d'une information en NRZ.

Certains supports autorisent une transmission directe des signaux numériques appelée *transmission en bande de base*. Elle conduit à des réalisations simples et économiques mais n'est pas possible sur tous les supports. De plus, pour une bonne transmission, la bande passante des signaux doit coïncider avec la bande passante du support. Lorsque ce n'est pas le cas, des techniques de *modulation* doivent être utilisées. Nous allons successivement détailler les techniques de transmission en bande de base et les transmissions par modulation.

3.1 Transmission en bande de base

Dans la transmission en bande de base, l'ETCD code le message de données synchrone en une suite de signaux compatibles avec les caractéristiques physiques du support de transmission (l'ETCD effectue, en fait, un simple transcodage du signal que fournit l'ETTD). Plusieurs facteurs expliquent les principales difficultés rencontrées dans la t*ransmission en bande de base* : la limitation de la bande passante – dans les basses comme dans les hautes fréquences – et le fait qu'il faille transférer les données quelle que soit leur valeur.



Nous verrons que les longues suites de 0 ou de 1 peuvent engendrer des problèmes à la réception.

Remarque

L'ETCD qui met en œuvre une transmission en bande de base est parfois appelé « modem bande de base » par abus de langage, bien qu'il ne fasse pas de modulation.

L'ETCD récepteur doit reconstituer correctement le signal d'horloge associé aux données. Pour cela, deux techniques de transmission de l'horloge sont envisageables : soit indépendamment du message de données (ce qui consomme une partie de la puissance disponible pour le signal), soit en utilisant les transitions du signal codé (il faut donc que le signal présente suffisamment de transitions). Dans ce dernier cas, si les données à transmettre contiennent une longue suite de 0 ou de 1, le signal NRZ reste à la même valeur pendant longtemps, provoquant ainsi une absence de repère temporel pour l'ETCD récepteur, d'où une perte de synchronisation. On ne transmet donc pas directement le signal en NRZ mais sous une forme voisine, qui prend en compte les contraintes précédentes. Le *code biphase* est un exemple très connu de codage pour la transmission des données en bande de base.

Le code biphase, également appelé code Manchester (voir figure 1.9), utilise une représentation à deux niveaux : pendant chaque intervalle de temps correspondant à un symbole binaire, deux polarités opposées sont transmises. Selon la donnée à coder, on trouve un front montant (transition vers le haut) ou un front descendant (transition vers le bas) au milieu de l'intervalle de temps significatif. Il y a donc systématiquement une transition du signal à chaque intervalle de temps, ce qui garantit une bonne synchronisation entre les deux ETCD et facilite le travail de décision du récepteur.

Remarque

Le code Manchester est le code le plus fréquemment employé dans les transmissions numériques. Il s'utilise en particulier dans les réseaux locaux Ethernet.

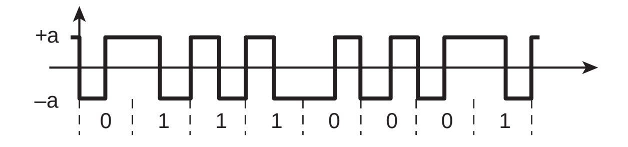


Figure 1.9
Code biphase ou Manchester.

3.2 Transmission par modulation

La transmission par modulation consiste à envoyer une onde sinusoïdale appelée *porteuse*. En fonction de la donnée à transmettre, l'ETCD modifie l'un des paramètres de la porteuse (fréquence, phase ou amplitude). Soit $a \cos(2\pi f_0 t + \phi)$ une porteuse de fréquence f_0, et $d(t)$ la suite des données binaires à transmettre (le message de données synchrone de la figure 1.7 par exemple). Appelons Δ l'intervalle de temps significatif pendant lequel $d(t)$ vaut 0 ou 1, c'est-à-dire que $d(t)$ est constant sur l'intervalle $[t, t + \Delta[$.

En *modulation d'amplitude simple*, l'amplitude du signal transmis change avec les données. Ainsi, pendant tout l'intervalle $[t, t + \Delta[$, le signal transmis vaudra : $m(t) = (a - k) \cos(2\pi f_0 t + \phi)$ si $d(t) = 0$, et $m(t) = (a + k) \cos(2\pi f_0 t + \phi)$ si $d(t) = 1$. Dans ces expressions,

k est une constante. À la réception, pendant l'intervalle $[t, t + \Delta[$, l'ETCD récepteur mesure l'amplitude du signal reçu et en déduit la valeur de la donnée *d(t)*.

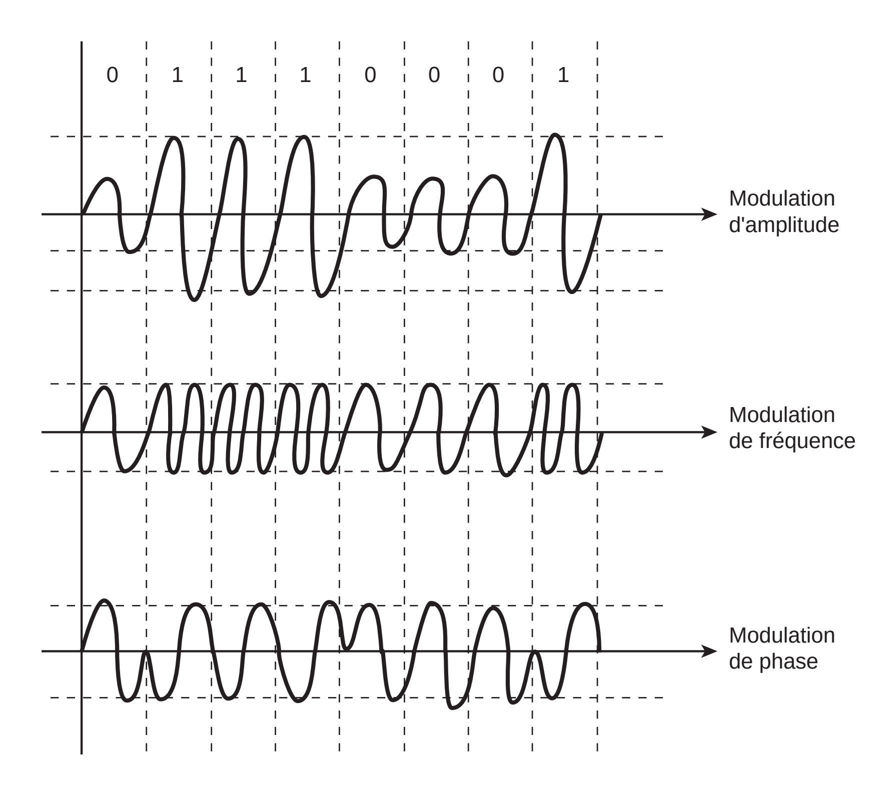
En *modulation de fréquence simple*, la fréquence du signal transmis change avec les données. Ainsi, pendant tout l'intervalle $[t, t+ \Delta[$, le signal transmis sera : $m(t) = a \cos(2\pi(f_0-h)t+\phi)$ si $d(t) = 0$ et $m(t) = a \cos(2\pi(f_0+\mathrm{h})t+\phi)$ si $d(t) = 1$, expressions dans lesquelles *h* est une constante. Pendant l'intervalle $[t, t+\Delta[$, l'ETCD récepteur mesure la fréquence du signal reçu et en déduit la valeur de la donnée *d(t)*.

En *modulation de phase simple*, la phase du signal transmis change avec les données. Ainsi, pendant tout l'intervalle $[t, t+ \Delta[$, le signal transmis sera : $m(t) = a \cos(2\pi f_0 t+\phi)$ si $d(t) = 0$ et $m(t) = a \cos(2\pi f_0 t+(\phi +\pi))$ si $d(t) = 1$. Pendant l'intervalle $[t, t+\Delta[$, l'ETCD récepteur mesure la phase du signal reçu et en déduit la valeur de la donnée *d(t)*.

Une modulation *simple* (voir figure 1.10) consiste à modifier la porteuse et à émettre le signal produit pendant l'intervalle Δ (qui dépend du débit binaire utilisé). Sur cet intervalle, la donnée à transmettre peut prendre deux valeurs (0 ou 1), et le signal aura deux valeurs (par exemple, les deux amplitudes $a - k$ et $a + k$). Le nombre de valeurs possibles du signal s'appelle la *valence* ; elle est notée *V*.

Figure 1.10

Exemples de modulations simples.



Pour atteindre des débits élevés, on pourrait imaginer de réduire l'intervalle Δ. Remplaçons Δ par $\Delta/3$: l'information *d(t)* change à chaque intervalle $\Delta/3$, de même que le signal modulé. Le récepteur n'a plus qu'un intervalle $\Delta/3$ pour effectuer ses mesures et prendre sa décision. Cette méthode devient peu fiable si on restreint trop l'intervalle de temps. On préfère donc découpler l'intervalle de variation des données, désormais noté θ, de l'intervalle de variation du signal modulé, toujours noté Δ ; on parle alors de modulation *complexe*. Par exemple, si θ vaut $\Delta/3$, les données contiennent 3 bits dans un intervalle Δ (donc huit valeurs différentes) : le signal modulé prend alors, pendant tout un intervalle Δ, une valeur parmi les 8 possibles.



4 Caractéristique d'une transmission

L'introduction d'une distance entre équipements informatiques nécessite un *support de transmission*. Or, nous avons vu que les ETCD cachaient la nature réelle du support à l'utilisateur (pour lequel elle est transparente). Celui-ci ne voit donc la transmission qu'à travers l'interface entre ETTD et ETCD. Du circuit de données, il ne connaît pratiquement que le débit binaire utilisé pour la transmission.

4.1 La qualité du circuit de données

La qualité du circuit de données est mesurée selon différents critères techniques :

- Le *taux d'erreurs* est le rapport entre le nombre de bits erronés, sur le nombre total de bits transmis.
- La *disponibilité* permet d'évaluer la proportion de temps pendant lequel la transmission est possible (absence de panne ou de coupure). On peut s'intéresser également au nombre d'incidents et à leur durée cumulée, afin de déterminer la durée moyenne et le coût d'une panne.
- Le *débit binaire D* représente le nombre de bits transmis par seconde. On peut préciser, en outre, que le débit est en mode duplex intégral, semi-duplex ou en simplex.
- La *rapidité de modulation R*, exprimée en bauds[3], indique le nombre de symboles transmis par unité de temps. Si Δ représente la durée (en secondes) de l'intervalle de temps séparant deux valeurs significatives du signal, alors $R = 1/\Delta$ bauds.
- Le *délai de propagation* définit le temps matériellement nécessaire au signal pour traverser le support. Par exemple, il faut environ un quart de seconde à un signal se propageant à la vitesse de la lumière pour parcourir une distance de 72 000 km (cas des satellites géostationnaires).

Remarque

La formule : $D = R^*\log_2 V$ exprime la relation liant la rapidité de modulation au débit binaire. Pour des modulations simples – des signaux de valence 2 – chaque intervalle Δ transporte 1 bit. Les valeurs numériques du débit binaire et de la rapidité de modulation sont alors égales.

Remarque

Pour un support de transmission, la rapidité de modulation maximale dépend de sa bande passante (critère de Nyquist). La rapidité de modulation maximale R_{max} est égale au double de la fréquence la plus élevée disponible sur le support : $R_{max} = 2F_{max}$.

4.2 Les données transmises

Les informations échangées sur le réseau proviennent de textes, de tableaux de nombres, d'images fixes ou animées, de musiques ou de sons : tout est mis sous forme numérique, c'est-à-dire de *données binaires*. La numérisation de la parole, du son, des images n'entre

3. Le mot *baud* vient d'Émile Baudot (1845-1903), ingénieur français.

pas dans le cadre de cet ouvrage. La notion de caractère (une lettre dans un texte) est elle aussi assimilée à une suite de bits (par exemple, chaque lettre ou chaque chiffre se code sur 7 bits dans l'alphabet ASCII. On peut donc représenter 2^7 soit 128 caractères différents avec ce code). D'une façon générale, on associe à tous les « objets » traités par l'informatique (et donc par les réseaux) des codes binaires dont la longueur dépend directement du nombre d'objets à dénombrer ou à coder.

Définition

En informatique, l'unité de quantité d'informations est le bit et tous ses multiples : octet, Kilo-octet (Ko), mégaoctet (Mo). Un Kilo-octet (avec un K majuscule)[4] contient 2^{10} octets, soit 1 024 octets (et non 1 000) ; un mégaoctet vaut 1 024 Kilo-octets soit 1 048 576 octets (et non 10^6…) ; les unités suivantes sont le gigaoctet (Go), qui vaut 1 024 Mo, le téraoctet (1 024 Go), le pétaoctet (1 024 To)…
Dans les réseaux informatiques et les télécommunications, le débit binaire s'exprime en bit/s et ses multiples : un kilobit/s (avec un k minuscule), un mégabit/s… ; ces dernières sont des puissances de 10 du bit/s. Ainsi, un modem à 56 kbit/s peut émettre ou recevoir jusqu'à 56 000 bit/s (et non 57 344 bit/s…).

4.3 L'interface série ETTD-ETCD

L'interface série entre l'ETTD et l'ETCD, ou *jonction*, est le point de raccordement physique entre l'équipement informatique et son modem. Les spécifications de cette jonction sont mécaniques (la forme du connecteur et le nombre de broches), électriques (les niveaux de tension utilisée) et fonctionnelles (signification des informations véhiculées sur chaque fil).

L'interface la plus courante est la jonction V24 (ou RS232C). Elle correspond à un connecteur ISO 2110 à 25 broches. Elle a une portée maximale de 50 m et un débit inférieur ou égal à 20 kbit/s. Le 1 est une tension négative comprise entre –3 V et –25 V, le 0 une tension positive comprise entre +3 V et + 25 V. À chaque broche correspond un fil (ou *circuit*). Chaque fil possède un numéro et joue un rôle dans l'échange de données comme le montre le tableau 1.2 dans lequel seuls figurent les principaux circuits.

Tableau 1.2

Les principaux circuits de l'interface série V24

Nº broche	Nº circuit	Rôle	Sens	Type
2	103	Données émises	ETTD vers ETCD	Données
3	104	Données reçues	ETCD vers ETTD	Données
4	105	Demande pour émettre	ETTD vers ETCD	Commande
5	106	Prêt à émettre	ETCD vers ETTD	Commande
6	107	Poste de données prêt	ETCD vers ETTD	Commande
8	109	Détection de porteuse	ETCD vers ETTD	Commande
20	108.2	Équipement de données prêt	ETTD vers ETCD	Commande

4. Bien que l'IEC (*International Electrotechnical Commission*) ait décidé d'aligner les unités d'informatique sur toutes les autres unités physiques avec kilo = 1 000, méga = 10^6…, de nombreux logiciels continuent d'utiliser les préfixes qui sont des puissances de 2.

L'initialisation d'un échange met en jeu les circuits 107 et 108.2. Lorsque l'ETTD veut émettre des données, il le signale par le circuit 105 pour que le modem se prépare (celui-ci envoie par exemple une porteuse non modulée pour que le modem distant se synchronise). Quand le modem est prêt, il répond sur le circuit 106. Enfin, les données à émettre sont fournies en série sur le circuit 103, accompagnées du signal d'horloge associé[5]. Lorsque le modem reçoit une porteuse de la part du modem distant, il le signale par le circuit 109. Dès qu'il est capable de démoduler le signal reçu, il en extrait les données qu'il transmet en série sur le circuit 104. L'ETTD échantillonne les données reçues grâce au signal d'horloge transmis par l'ETCD. L'échange de données sur les circuits 104 et 105 peut avoir lieu simultanément si la transmission est en duplex intégral.

On voit donc qu'un dialogue existe entre l'ETTD et l'ETCD, par l'intermédiaire des différents circuits. La normalisation définit ce dialogue indépendamment du mode de transmission, du support utilisé et de la configuration du circuit. L'interface série *V24* a été très largement répandue, en particulier avec un connecteur simplifié à 9 broches seulement (*DB9* au lieu de *DB25*). Elle a fait place à d'autres interfaces plus performantes, comme RS449 qui peut supporter jusqu'à 2 Mbit/s. Depuis 1995, on utilise fréquemment le port USB (*Universal Serial Bus*) – dont la version la plus rapide supporte jusqu'à 480 Mbit/s – car il permet de brancher ou de débrancher le modem à chaud (sans avoir à redémarrer l'ordinateur). Le port USB ne contient que 4 circuits : un pour l'alimentation, deux pour les données (un par sens de transmission) et une terre de protection. Le dialogue de l'interface se déroule alors directement sur les circuits de données, par des échanges de messages codés.

5 ADSL (*Asymmetric Digital Subscriber Line*)

L'ADSL fait partie d'une famille d'accès à haut débit (d'où le nom générique *xDSL* donné à ces techniques de transmission), qui utilise les lignes téléphoniques ordinaires comme support de transmission. L'ADSL utilise la *boucle locale* raccordant chaque usager du téléphone au central téléphonique dont il dépend. L'idée est la suivante : puisque la bande passante utilisée pour les conversations téléphoniques est faible (de 300 à 3 400 Hz), la majeure partie de la bande passante des paires torsadées est inutilisée et peut s'employer pour le transfert des données numériques. L'ADSL multiplexe, sur la ligne de l'abonné, les données numériques (provenant d'un ordinateur par exemple) et le téléphone vocal. Les deux équipements s'utilisent ainsi simultanément sans interférences.

Une des caractéristiques de l'ADSL tient dans son nom : le débit est différent dans les deux sens de transmission ; le sens le moins rapide possède un débit environ 10 fois inférieur à l'autre sens. Le débit binaire disponible dépend de la longueur et de l'état de la boucle locale. Ces deux facteurs déterminent, à l'initialisation, le débit maximal offert à l'abonné. Au départ, l'ADSL permettait d'émettre jusqu'à 8 Mbit/s dans le sens *descendant* (du fournisseur vers l'usager) et jusqu'à 800 kbit/s dans le sens *montant* (de l'usager vers le fournisseur). Les dernières versions offrent des débits pouvant aller jusqu'à 50 Mbit/s, mais sur des distances beaucoup plus courtes.

Dans le central téléphonique, les deux types de systèmes coexistent : le réseau de données (le réseau du fournisseur d'accès) vient se greffer sur le réseau téléphonique classique, les

5. Le circuit utilisé pour l'horloge dépend du type du modem et de la nature du transfert de données (émission ou réception) : une émission utilise le circuit 113 ou 114, une réception utilise le circuit 115. Quand la transmission est en mode duplex intégral, le modem utilise deux signaux d'horloge, un par sens de transmission.

deux réseaux utilisant la ligne de l'abonné (voir figure 1.11). Les deux types de signaux sont acheminés dans leurs équipements respectifs, chez l'abonné comme dans le central téléphonique. Un équipement appelé *répartiteur* (*splitter*) est responsable de l'éclatement et de la recombinaison des deux types de signaux dans le central et chez l'abonné (indispensable chez ce dernier uniquement lorsque celui-ci utilise un téléphone numérique ; il sert alors à séparer les canaux utilisés pour la téléphonie de ceux employés pour la transmission des données). Pour un téléphone analogique, un simple filtre placé devant le téléphone de l'abonné suffit.

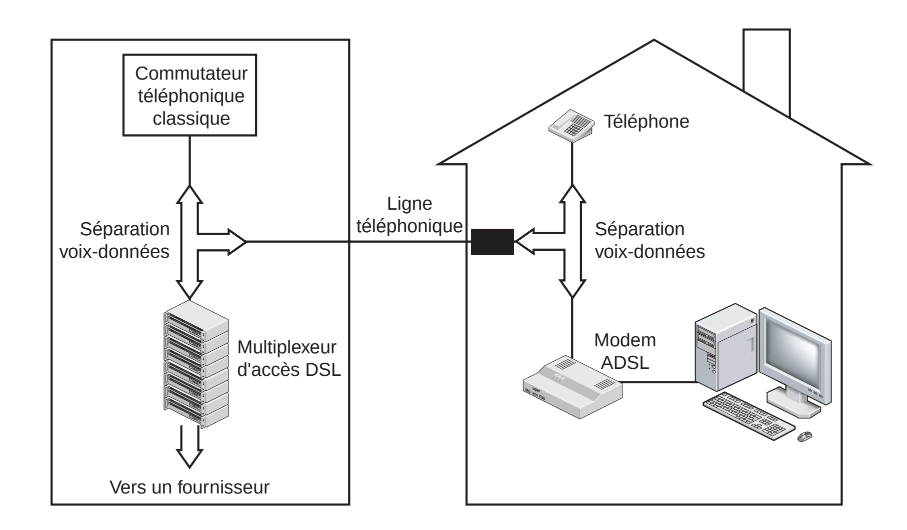


Figure 1.11

Raccordement ADSL.

La transmission des données de l'ADSL utilise une modulation particulière (DMT, *Discrete MultiTone*), spécifiquement adaptée aux caractéristiques physiques des lignes d'abonnés sur une courte distance (généralement moins de 3,5 km) et utilisant deux débits différents. Le modem ADSL logé chez l'abonné et l'interface utilisateur peuvent se présenter sous plusieurs formes, dont la plus récente est le port USB.

Résumé

Pour relier deux équipements informatiques éloignés l'un de l'autre, on utilise un circuit de données constitué par un support de transmission, des modems et une interface série.

Les supports de transmission sont très variés (paires métalliques, câbles coaxiaux, fibre optique, sans fil…). La bande passante et le taux d'erreur qu'il introduit dans les signaux transportés sont les principales caractéristiques d'un support. À chaque extrémité, des modems (modulateurs-démodulateurs de signaux analogiques) ou des codecs (codeurs-décodeurs de signaux numériques) transmettent des signaux adaptés à la nature du support. Les techniques de transmission de données (en bande de base ou par modulation) permettent d'adapter au mieux les signaux aux caractéristiques des supports. Une interface série relie chaque modem à l'équipement informatique qui envoie ou reçoit des données. Les techniques et les interfaces sont normalisées au niveau international par l'ISO et l'ITU.

Le raccordement ADSL des usagers à Internet est un exemple de transmission utilisant la boucle locale téléphonique. Un multiplexage de la téléphonie et des données utilise une modulation spécifique. L'interface série est fréquemment un port USB.

Problèmes et exercices

Exercice 1 La notion de décibel

Énoncé

Dans un environnement urbain, la puissance sonore produite par les nombreuses sources de bruits est évaluée en décibels, en comparant la puissance sonore de la source de bruit à un niveau sonore de référence.

a Si on évalue la puissance sonore *S* d'une grosse moto à 87 dB, quelle est, en décibels, la puissance sonore produite par une bande de 8 motards roulant sur des motos identiques circulant à la même vitesse ?

b Trouvez la puissance sonore réellement émise.

Solution

a La bande de motards produit 8 fois plus de puissance sonore qu'une seule moto. On a donc : $10*\log_{10}(8S) = 10*\log_{10} 8 + 10*\log_{10} S$, ce qui revient à *ajouter* 10 fois le logarithme décimal de 8 au bruit d'une moto pour obtenir le nombre de décibels produit par les 8 motos.

Puisque : $10*\log_{10}8 = 10*\log_{10}2^3 = 3*10*\log_{10}2 = 9$ dB, la puissance des 8 motos vaut : $S = 87 + 9 = 96$ dB.

b Cela correspond à une puissance sonore de $4*10^9$, soit 4 milliards de fois le fond sonore de référence !

Remarque

Pendant que la valeur en décibels du bruit a augmenté d'environ 10 %, la puissance sonore réellement émise a été multipliée par 8.

Exercice 2 Évaluation d'un rapport signal/bruit (S/B)

Énoncé

Sur un support de transmission, le rapport *S/B* vaut 400.

a Quelle est la valeur de ce rapport en décibels ?

b Même question avec un rapport *S/B* de 40 000.

c Quelle est la valeur *N* en décibels d'un rapport *S/B* égal à 500 000 ?

Solution

a Un rapport S/B de 400 correspond à $10*\log_{10}400 : 10*(\log_{10}4 + \log_{10}100)$.
D'où : $20*(\log_{10}2 + \log_{10}100) = 26$ dB.

b Le rapport S/B est 100 fois plus élevé que le précédent, c'est-à-dire qu'il vaut : $26 + 20 = 46$ dB.

c On peut calculer simplement une bonne valeur approchée du nombre *N* de décibels en remarquant que : $500\,000 = 10^6 \div 2$. On aura donc :
$N = 10*(\log_{10}10^6 - \log_{10}2) = 10*[6*\log_{10}10 - \log_{10}2] = 60 - 3 = 57$ dB.

EXERCICE 3 DÉBIT BINAIRE ET RAPIDITÉ DE MODULATION

Énoncé

Soit un signal numérique dont la rapidité de modulation est 4 fois plus faible que le débit binaire.

a Quelle est la valence du signal ?

b Si la rapidité de modulation du signal vaut 2 400 bauds, quel est le débit binaire disponible ?

Solution

a D'après la formule $D = R\ log_2 V$, nous trouvons : $D/R = log_2 V$ soit : $V = 2^{D/R}$, c'est-à-dire que la valence vaut 16.

b En appliquant la même formule, nous trouvons : D = 2 400*4 = 9 600 bit/s.

EXERCICE 4 SIGNAUX TRANSMIS EN BANDE DE BASE ET PAR MODULATION

Énoncé

Soit la suite d'éléments binaires `0 1 1 1 1 1 1 0`.

a Représentez les signaux transmis lorsqu'on transmet en bande de base avec les codes NRZ et Manchester.

b Représentez les signaux transmis lorsqu'on transmet les données avec une modulation d'amplitude à deux valeurs, une modulation de phase à deux valeurs, une modulation de fréquence à deux valeurs.

c Si le débit D est connu, quelle est la rapidité de modulation R ?

Solution

a Les figures 1.12 et 1.13 représentent les données codées en NRZ et Manchester :

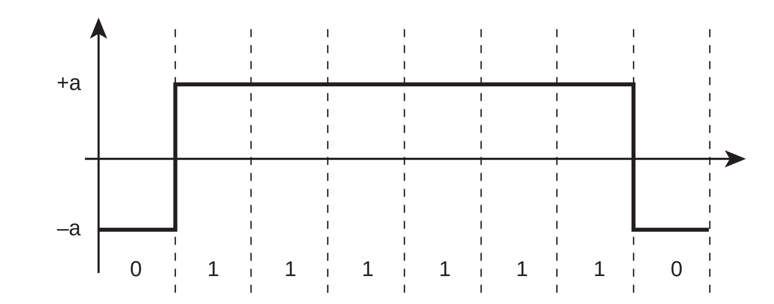


Figure 1.12
Codage NRZ.

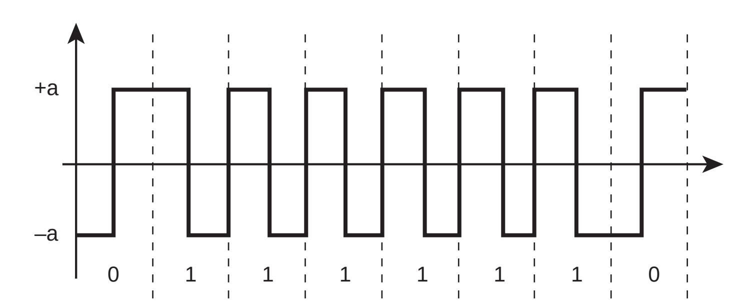


Figure 1.13
Codage biphase ou Manchester.

b Les modulations d'amplitude, de fréquence et de phase sont représentées à la figure 1.14.

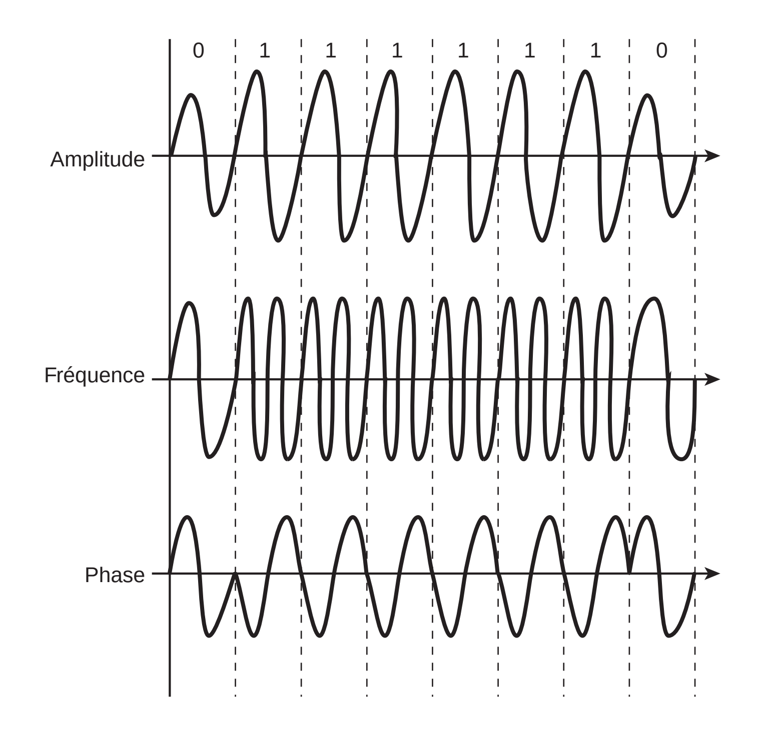


Figure 1.14

Représentation des différentes modulations.

c Si D est connu et que la valence des signaux est égale à 2, alors $R = D$ bauds.

EXERCICE 5 CODE MANCHESTER ET AUTRES CODES

Énoncé

Le code Manchester présente l'intérêt de posséder au moins une transition du signal au milieu de l'intervalle pour une bonne synchronisation du récepteur mais il peut présenter trop de transitions, en particulier si la suite de données binaires contient une longue suite de 0 par exemple.

a Représentez le signal transmis avec le code Manchester pour les données `100000000001`.

b Le code de Miller offre une alternative intéressante. Il consiste, à partir du code Manchester, à supprimer une transition sur deux. Dessinez le signal transmis pour les mêmes données et montrez que le décodage n'est pas ambigu.

Solution

a La figure 1.15 représente les données avec le code Manchester.

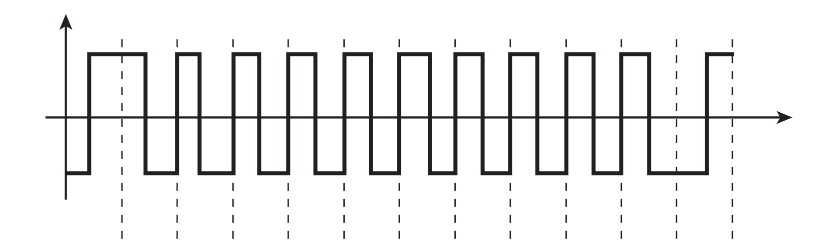


Figure 1.15

Données en codage Manchester.

b La figure 1.16 représente les données avec le code de Miller.

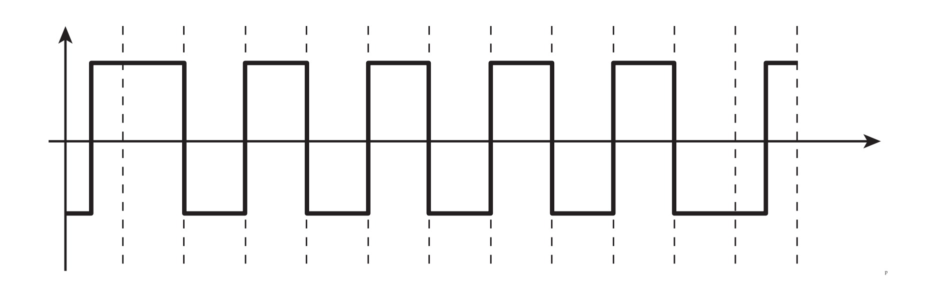


Figure 1.16
Données en codage de Miller.

Le décodage du code de Miller est très simple : une transition au milieu de l'intervalle représente un 1, une absence de transition dans l'intervalle représente un 0. Il n'y a donc aucune ambiguïté de décodage.

Remarque

Le choix d'un « bon » code est difficile ! Il faut trouver un compromis entre le nombre de transitions indispensable à la synchronisation du codec récepteur et une solution transparente aux données transmises.

EXERCICE 6 FORMULE DE SHANNON

Énoncé

Si on n'utilise pas de techniques de compression de données, une transmission de voix numérisée nécessite un débit binaire de 64 kbit/s.

a En supposant que la transmission se fasse par des signaux modulés de valence 32, quelle est la bande passante disponible, sachant que celle-ci est égale à la moitié de la rapidité de modulation utilisée ?

b Quel doit être le rapport S/B de la ligne de transmission offrant un débit binaire de 64 kbit/s et possédant une largeur de bande trouvée dans la question précédente ? On exprimera cette valeur en vraie grandeur et en décibels.

Solution

a On utilise la formule $D = R{*}log_2V$.

On obtient : $64{*}10^3 = R{*}\log_2 32$, ce qui donne $D = 5R$, d'où : $R = 12\,800$ bauds. La bande passante est donc égale à 6 400 Hz.

b En utilisant la formule de Shannon $D = W{*}log_2(1 + S/B)$, on trouve : $64{*}10^3 = 6\,400{*}\log_2(1 + S/B)$, d'où : $\log_2(1 + S/B) = 10$, c'est-à-dire que $S/B = 2^{10} - 1$, soit 1 023 (on pourra négliger le 1 devant le rapport *S/B*), ce qui correspond à 30 dB environ.

EXERCICE 7 CONNEXION À INTERNET

Énoncé

Pour vous connecter à Internet, vous avez relié votre ordinateur portable au réseau grâce à un modem de type PCMCIA, raccordé à la ligne téléphonique de votre domicile. On suppose que votre modem a un débit maximal de 56 kbit/s et que votre ligne téléphonique possède une bande passante comprise entre 300 et 3 400 Hz. Pendant votre connexion, vous constatez que la vitesse de transfert des données effective est 6 200 octet/s.

a Si la vitesse constatée ne provient que d'un mauvais rapport *S/B* de votre ligne, quelle est la valeur de ce rapport durant votre connexion ?

b La vitesse de transmission est maintenant de 24 800 bit/s. Si la rapidité de modulation est de 4 800 bauds, quelle est la valence du signal modulé ?

c On suppose que la ligne téléphonique répond au critère de Nyquist et que la rapidité de modulation vaut 4 800 bauds. Si on utilise la rapidité de modulation maximale, quelle est la bande passante du support ?

d Supposons que le débit binaire indiqué reste constant et égal à 49 600 bit/s pendant toute la durée de la connexion. Combien de temps devrez-vous rester connecté pour télécharger un fichier de 2 Mo (on pourra prendre ici 1 Mo = 10^6 octets) sur votre portable ?

e Vous utilisez désormais une connexion à 10 Mbit/s. Combien de temps resterez-vous connecté pour télécharger le même fichier que celui de la question d ?

Solution

a Le débit binaire de la ligne vaut 49 600 bit/s. D'après le théorème de Shannon, on obtient : 49 600 = 3100*$\log_2(1 + S/B)$, soit : $\log_2(1 + S/B) = 16$, d'où : $S/B = 2^{16} - 1$. En négligeant le 1, nous trouvons un rapport *S/B* = 65536, soit environ 48 dB.

b Toujours en utilisant le théorème de Shannon, nous trouvons : 24 800 = 3100*$\log_2(1 + S/B)$, soit : $S/B = 2^8 - 1 = 255$. Le rapport *S/B* vaut environ 24 dB.

c Selon le critère de Nyquist, la rapidité de modulation maximale est égale à 2 fois la bande passante de la ligne. Celle-ci vaut donc 2 400 Hz.

d Le temps t nécessaire pour transférer 2*10^6 octets est égal à : t = 2*8*10^6/49 600 = 322,58 s soit environ 5 minutes et 22 secondes.

e Le temps t nécessaire n'est plus que de 1,6 s…

EXERCICE 8 CARACTÉRISTIQUES DES MODEMS V23 ET V29

Énoncé

a Exprimez et comparez les valeurs du débit binaire et de la rapidité de modulation du modem *V23* et du modem *V29*. Le modem *V23* fonctionne à 1 200 bit/s avec une modulation de fréquences à deux valeurs. Le modem *V29* offre un débit binaire de 9 600 bit/s et utilise une modulation combinée d'amplitude et de phase (modulation d'amplitude à 2 valeurs et modulation de phase à 8 valeurs).

b Proposez un codage simple des données binaires transmises par le modem *V29*.

Solution **a** Le modem normalisé *V23* est le « vieux » modem intégré au Minitel. Les caractéristiques techniques fournies montrent qu'il transmet des signaux de valence 2, c'est-à-dire qu'un intervalle de temps de 1/1 200 s contient 1 bit. Donc la rapidité de modulation de ce modem est égale à son débit binaire soit 1 200 bauds.

Dans le modem V29, on utilise deux amplitudes *A1* et *A2* et huit phases *P1*, *P2*, *P3*, *P4*, *P5*, *P6*, *P7*, *P8*. Pendant un intervalle de temps, il s'agit de la combinaison d'une amplitude et d'une phase, donc le modem transmet une valeur parmi les 16 possibles. Il transmet 4 bits par intervalle de temps ; les informations à transmettre sont codées par groupes par 4 bits (appelés parfois quadribits) par le modem. Voici un exemple possible de codage des quadribits :

`0000` ==> *A1* et *P1*	`1000` ==> *A2* et *P1*
`0001` ==> *A1 et P2*	`1001` ==> *A2* et *P2*
`0010` ==> *A1* et *P3*	`1010` ==> *A2* et *P3*
`0011` ==> *A1* et *P4*	`1011` ==> *A2* et *P4*
`0100` ==> *A1* et *P5*	`1100` ==> *A2* et *P5*
`0101` ==> *A1* et *P6*	`1101` ==> *A2* et *P6*
`0110` ==> *A1* et *P7*	`1110` ==> *A2* et *P7*
`0111` ==> *A1* et *P8*	`1111` ==> *A2* et *P8*

b Comme le débit du modem *V29* est de 9 600 bit/s, l'intervalle de temps est de 4/9 600 s, soit 1/2 400 s ; la rapidité de modulation vaut : 9 600/4 = 2 400 bauds. On peut retrouver ce résultat en appliquant la formule : $D = R*log_2V$, dans laquelle *D* et *V* sont connus et valent respectivement 9600 et 16.

EXERCICE 9 MODEM NORMALISÉ V32

Énoncé Vous avez déniché dans votre cave un vieux modem fonctionnant à l'alternat, capable d'envoyer et de recevoir les données à 9 600 bit/s. Sans connaître les normes utilisées dans la construction de ce modem, vous essayez de trouver ce que pourraient être sa rapidité de modulation et la valence des signaux qu'il produit, sachant que la bande passante du téléphone vaut 3 100 Hz. Indiquez les solutions que vous avez trouvées.

Solution La seule chose certaine est que la valence du signal produit doit être supérieure à 2 puisque, d'après le critère de Nyquist, le modem ne pourrait envoyer (ou recevoir) que 6 200 bit/s au maximum avec cette valence.

En appliquant la formule liant la rapidité de modulation au débit binaire et à la valence, vous obtenez : $\log_2 V = 9600/3100$, soit $V = 2^{3{,}097}$ environ. Sans même faire le calcul, vous vous rendez compte que cette solution est inacceptable puisque, par définition, la valence est un nombre entier. D'autre part, le débit binaire ne vaudra pas *exactement* 9 600 bit/s ! Il faut donc que la rapidité de modulation soit un sous-multiple entier du débit binaire, c'est-à-dire que le rapport entre les deux grandeurs doit être une puissance de 2.



Vous proposez :

a. Une rapidité de modulation valant 2 400 bauds. D'où : $D = 4R$ et donc $V = 16$.

b. Une rapidité de modulation valant 1 200 bauds. D'où : $D = 8R$ et donc V = 256.

c. Une rapidité de modulation valant 3 200 bauds. D'où : $D = 3R$ et donc V = 8.

En fouillant dans les vieilles normes AFNOR (Association française de normalisation), vous constatez que les modems transmettant à l'alternat à 9 600 bit/s fonctionnaient conformément à la recommandation *V32*. Cette norme préconisait une rapidité de modulation de 2 400 bauds ; la modulation employée était une modulation d'amplitude complexe utilisant une valence 16 ou 32.

Remarque

La valence 32 fournit ici un débit binaire de 5*2 400 = 12 000 bit/s soit plus que 9 600 ! En fait, sur les 12 000 bits transmis à la seconde, seuls 9 600 étaient réellement « utiles », les 2 400 autres servant de protection contre les erreurs : ils étaient calculés à partir des 9 600 premiers et permettaient au modem récepteur de détecter et de corriger d'éventuelles erreurs. Pour l'utilisateur, le débit réellement utilisable reste de 9 600 bit/s. Ce mécanisme, très général, sera repris au chapitre 2.

Exercice 10 Système de radiomessagerie

Énoncé

Un système de radiomessagerie de poche (un *pager*) répondant à la norme ERMES (*European Radio Message System*) présente les caractéristiques techniques suivantes :

- bande de fréquences : 169,425 MHz – 169,800 MHz ;
- modulation de fréquences à 4 états ;
- rapidité de modulation : 3 125 bauds ;
- rapport *S/B* d'un récepteur : 76 dB.

a Quel est le débit binaire réellement utilisé dans cette radiomessagerie ?

b En supposant qu'on transmette un octet par caractère, combien de temps faut-il pour transmettre un message de 200 caractères sur un récepteur de radiomessagerie ?

c Au lieu du débit binaire trouvé à la question a, quel débit binaire pourrait-on théoriquement obtenir en exploitant au mieux les caractéristiques techniques de la radiomessagerie ?

d Pourquoi n'est-ce pas utilisé ?

Solution

a Le débit binaire réellement utilisé est : D = 3 125*2 = 6 250 bit/s.

b Il faut : 8*200/6250 = 0,256 seconde pour transférer le message sur le récepteur.

c La bande passante du support vaut : (169,8 – 169,425)*10^6 = 375 kHz. D'après le théorème de Shannon, on pourrait transmettre au maximum : $D = 375*10^3*\log_2(1 + S/B)$ soit environ : 9 467 495 bit/s.

d Parce que la vitesse d'affichage utilisée est bien suffisante pour un lecteur humain, puisqu'un écran entier s'affiche en un quart de seconde. On peut ainsi se contenter d'employer des composants bon marché pour la fabrication des récepteurs.

EXERCICE 11 CODAGE DES INFORMATIONS

Énoncé

On utilise un alphabet de 26 caractères différents. Pour transmettre ces données, on code chaque caractère par une suite de bits.

a Si le codage des caractères est à longueur constante, combien faut-il de bits pour coder un caractère de cet alphabet ?

b Dans le réseau Télex (réseau télégraphique), on utilisait un alphabet contenant 5 bits par caractère. Comment pouvait-on coder les lettres de l'alphabet latin, les chiffres et d'autres symboles (comme les signes de ponctuation, par exemple) ?

c Combien de caractères différents peut-on représenter avec la méthode de codage précédente ?

Solution

a Il faut coder chaque caractère de l'alphabet avec un nombre constant de bits. 26 étant un nombre compris entre 16 et 32, on choisira donc la puissance de 2 par excès qui permet de coder tous les caractères, même si certains codages sont inutilisés. Il faut donc $\log_2 32$ bits pour coder les caractères de l'alphabet, c'est-à-dire 5 bits.

b Avec 5 bits, on peut coder 2^5 symboles soit 32 caractères différents, ce qui est notoirement insuffisant pour coder les 26 lettres, plus les chiffres et les signes de ponctuation. Les télégraphistes ont donc inventé la notion de *caractère d'échappement*, un caractère dont la présence modifie l'interprétation du ou des caractères qui suivent. On a défini un code « Lettre » et un code « Chiffre », les autres caractères étant interprétés en fonction du caractère d'échappement qui les précède. Ainsi, chaque codage binaire a deux interprétations, selon qu'on se trouve en mode « Lettre » ou en mode « Chiffre » (par convention, on reste dans le mode sélectionné tant qu'on ne trouve pas un nouveau caractère d'échappement).

c On dispose ainsi de 30 caractères en configuration « Lettre » et 30 caractères en configuration « Chiffre ». On dispose donc au maximum de 62 codes différents pour représenter tous les caractères.

Remarque

Pour la saisie des caractères dactylographiés sur un clavier, la touche Maj est un caractère d'échappement qui modifie la valeur du caractère suivant : tant qu'on n'appuie pas sur cette touche, on tape les minuscules ou les caractères spéciaux correspondant au codage en mode Minuscule.

Exercice 12 Interface ETTD-ETCD

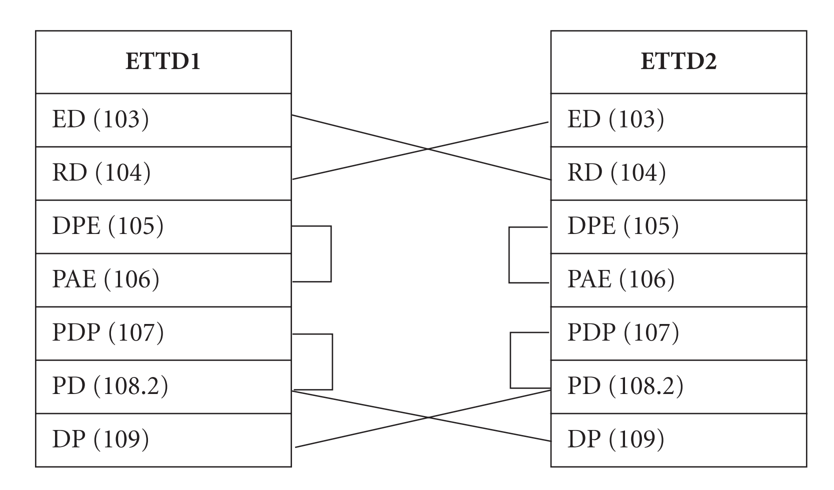
Énoncé

L'interface ETTD-ETCD définie par *V24* est conçue de telle sorte qu'un ETTD ne puisse être relié qu'à un ETCD. Peut-on relier deux ETTD utilisant localement (c'est-à-dire sans ETCD) une interface *V24* ? Reliez les signaux des circuits des deux ETTD donnés sur la figure pour permettre un échange de données correct entre les deux ETTD.

ETTD1	**ETTD2**
ED (103)	ED (103)
RD (104)	RD (104)
DPE (105)	DPE (105)
PAE (106)	PAE (106)
PDP (107)	PDP (107)
PD (108.2)	PD (108.2)
DP (109)	DP (109)

Solution

Oui, on appelle une telle liaison « zéro-modem » (ou *null modem*). Cela consiste à croiser les fils, de telle sorte que ce qui est émis par l'ETTD1 soit reçu par l'ETTD2 et *vice versa*. Les ETCD étant absents, le câblage doit toujours être prêt à émettre, d'où la boucle locale des circuits 105 – 106 de chaque côté, de même pour les circuits 107 et 108. Enfin, il n'y a plus de porteuse mais les deux ETTD doivent être informés du fonctionnement de l'interface. Les circuits 107 et 109 doivent recevoir un signal pendant la durée des échanges : on utilise pour cela le signal 108.



EXERCICE 13 PRINCIPES DE FONCTIONNEMENT DE L'ADSL

Énoncé

Examinons les principes de transmission utilisés dans l'ADSL. Dans la modulation DMT, la plage des fréquences disponible sur la boucle locale est divisée en 256 canaux juxtaposés de 4 312,5 Hz chacun. Le canal 0 est utilisé pour le téléphone vocal et les canaux 1 à 5 ne sont pas exploités pour éviter les interférences entre la voix et les données. Parmi les canaux restants, deux sont réservés pour le contrôle des flux montant et descendant, le reste est utilisé pour transmettre les données.

a Combien reste-t-il de canaux à utiliser pour le transfert des données dans les deux sens en modulation DMT ?

b De quoi dépend le nombre de canaux à affecter aux données de chaque sens ? Qui se charge de l'affectation des canaux ?

c Que faudrait-il faire pour que les flux montant et descendant aient des débits identiques ?

d L'utilisation la plus courante en ADSL consiste à réserver 32 canaux pour le flux montant et les canaux restants pour le flux descendant. Quel est le débit théorique que l'on peut obtenir pour le flux montant si l'on transmet des signaux binaires sur chaque canal ?

e Même question pour le flux descendant.

f Une autre technique de modulation utilise, pour le flux descendant, une rapidité de modulation de 4 000 bauds et émet 15 bits par signal transmis sur 224 canaux. Quel débit binaire peut-on obtenir avec cette technique ?

Solution

a Il reste 248 canaux pour les flux de données montant et descendant.

b Le nombre de canaux affectés à chaque sens dépend du débit binaire qu'on veut offrir aux abonnés : plus ce nombre est grand et plus le débit binaire sera important pour le flux considéré. C'est bien évidemment le fournisseur d'accès qui répartit les canaux, en allouant généralement 90 % des canaux au flux descendant et les 10 % restants au flux montant.

c Il faut simplement allouer autant de canaux pour le flux montant que pour le flux descendant. On obtient ainsi une technologie DSL symétrique (SDSL).

d On peut obtenir : 4 312,5*32 = 138 kbit/s pour le flux montant.

e Il reste pour le flux descendant : 248 – 32 = 216 canaux, soit un débit binaire de 931,5 kbit/s.

f On peut obtenir : 15*4 000*224 = 13,44 Mbit/s.

Remarque

Les technologies symétriques sont réservées aux opérateurs et aux fournisseurs d'accès. Elles ne sont pas disponibles pour les abonnés. On n'atteint pas dans la pratique le débit obtenu à la question f, car le rapport *S/B* des lignes est insuffisant le plus souvent. On obtient couramment 8 Mbit/s sur de courtes distances, avec une boucle locale de bonne qualité.



Les protocoles de liaison de données

Le circuit de données pouvant altérer les informations transportées, le *protocole de liaison de données* le supervise et définit un ensemble de règles pour assurer la fiabilité des échanges sur une *liaison de données*.

Ce protocole spécifie le format des unités de données échangées (les trames), leur délimitation, les moyens de contrôler leur validité (parité, code polynomial…), ainsi que le mode de correction des erreurs détectées. Il fixe également les règles du dialogue entre les deux extrémités de la liaison. Il exerce en outre deux fonctions importantes : le contrôle de flux (mécanisme vérifiant le rythme d'envoi des informations) et la gestion des acquittements (mécanisme validant la réception des informations).

HDLC (*High level Data Link Control*) est un exemple de protocole normalisé très répandu, orienté bit, transparent à tous les codes, dans lequel toutes les trames ont le même format. Il permet d'exploiter une liaison bidirectionnelle simultanée avec contrôle d'erreurs, de séquence et de flux.

PPP (*Point to Point Protocol*) en est une version très simplifiée, utilisée dans Internet.

1 Rôle et fonctions d'un protocole de liaison

Pour faire communiquer des machines identifiées par leurs adresses, il faut définir un grand nombre de règles concernant la structuration du dialogue, le format des messages transmis, leur enchaînement logique, le codage de l'information, le rythme de transmission, etc. L'ensemble des règles, assimilables à des règles d'orthographe et de grammaire définissant la construction des phrases d'une langue, s'appelle *protocole de liaison de données* ou *protocole de communication*. Un programme (*logiciel de communication*), installé sur les équipements qui doivent communiquer à distance, l'exécute. Afin d'assurer un maximum d'interopérabilité entre équipements différents, les instances de normalisation ont travaillé à la définition des protocoles de communication à l'échelle internationale.

Définition

Un *protocole* est un ensemble de règles et de formats de données à respecter pour échanger des données dans de bonnes conditions entre deux équipements ou deux programmes. Un protocole *de liaison de données* a pour objet de rendre fiable le circuit de données.

Alors que le circuit de données transmet des éléments binaires, le protocole de liaison de données travaille sur des blocs d'éléments binaires appelés *trames*. La trame est donc l'unité de données qu'il gère. Elle transporte les données de l'utilisateur et contient, en outre, des informations de commande, nécessaires au protocole pour garantir le bon déroulement du dialogue (certaines trames, les trames de *supervision*, sont d'ailleurs réduites aux seules informations de commande). Une trame compte différents *champs*. Chacun d'eux est un bloc d'éléments binaires dont la signification et l'interprétation sont précisées dans la définition du protocole.

Le protocole doit également définir les règles du dialogue et spécifier la façon de corriger les erreurs détectées. Enfin, on doit pouvoir détecter les pannes des équipements ou les ruptures complètes de liaison pour avertir l'utilisateur de l'indisponibilité du service.

Remarque

Définir un protocole de liaison de données consiste à préciser : le format des trames échangées, les conditions de délimitation des trames (début et fin) et leur validité, la position et la signification des différents champs d'une trame, la technique de détection d'erreur utilisée, les règles du dialogue (supervision de la liaison) et les procédures à respecter après détection d'erreurs ou de panne de la liaison.

1.1 Mise en forme des données

En théorie, les délimitations de début et de fin de trame sont indépendantes de la technique de transmission utilisée. En pratique, certains procédés utilisent des particularités du codage en ligne pour délimiter les trames. Les solutions les plus fréquentes sont la délimitation par une séquence binaire spéciale ou l'indication explicite de la longueur de la trame.

Délimitation par une séquence spécifique d'éléments binaires

Les trames ayant un nombre quelconque de bits, une séquence spécifique, appelée *fanion* (ou *flag*), sert à indiquer le début aussi bien que la fin des trames. En général, il se compose de l'octet `01111110`. Un mécanisme de transparence est nécessaire pour éviter de retrouver cette séquence à l'intérieur d'une trame : à l'émission, on insère dans le corps de la trame un élément binaire 0 après avoir rencontré cinq éléments binaires consécutifs de valeur 1.



En réception, il faut supprimer l'élément binaire de valeur 0 après avoir rencontré cinq éléments binaires consécutifs de valeur 1. Un tel mécanisme (le *bit stuffing*) interdit l'émission de plus de cinq éléments binaires de valeur 1 dans le corps de la trame, puisque cette configuration est réservée à sa délimitation. Cette méthode permet la transmission de trames de longueur quelconque sans contraintes particulières.

Exemple

Prenons les données utiles suivantes : `0110 1111 1110 1001`. Précédées et suivies de fanions, elles seront réellement émises sous la forme : `01111110` `0110 1111 10110 1001` `01111110`. Dans cette séquence, les fanions sont soulignés et le bit inséré pour la transparence est en gras souligné.

Délimitation par transmission de la longueur du champ de données

Une autre méthode de délimitation consiste à indiquer, dans un champ particulier, le nombre d'octets utiles contenus dans la trame. Après une séquence de début de trame, un ou plusieurs octets indiquent sa longueur (l'emplacement de ce champ est fixé par rapport au début de trame), qui s'exprime généralement en octets ou en nombre de mots (de 16 ou 32 bits, par exemple).

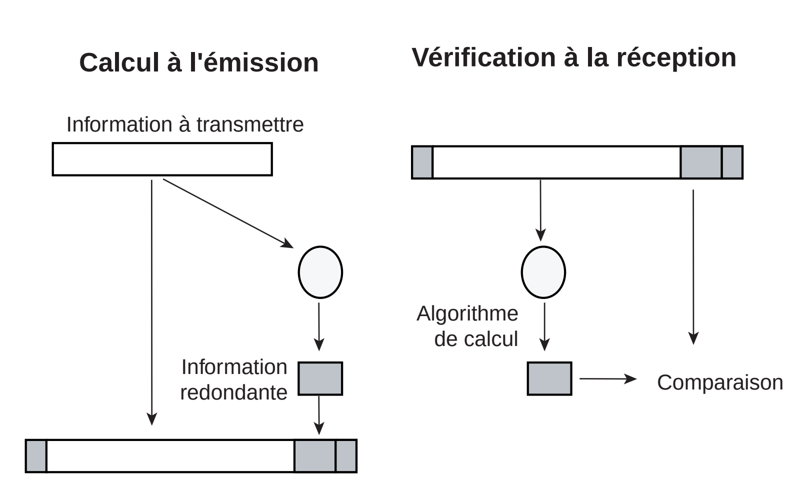
Ce procédé induit une limitation de la taille des trames : ainsi, si la longueur, exprimée en octets, est codée sur un octet, on se limite à des trames de 256 octets. Ce faisant, on évite les problèmes de transparence puisque le récepteur n'interprète en aucun cas les données reçues comme des délimiteurs. La longueur de la trame peut être ajustée à une longueur fixe par ajout d'éléments binaires de remplissage. Le champ précise alors la taille des données utiles transportées dans la trame.

1.2 Contrôle de la validité de l'information transmise

Le *contrôle d'erreurs* consiste à vérifier la validité des données transmises. Si on admet que le service de transmission n'est pas fiable, il faut se protéger contre d'éventuelles erreurs, donc les détecter puis les corriger. Pour cela, on ajoute à l'information transmise une *redondance*, c'est-à-dire des informations de contrôle calculées par un algorithme spécifié dans le protocole à partir du bloc de données. À la réception, on exécute le même algorithme pour vérifier si la redondance est cohérente. Si c'est le cas, on considère qu'il n'y a pas d'erreur de transmission et l'information reçue est traitée ; sinon, on est certain que l'information est invalide et elle est ignorée. La figure 2.1 illustre le principe de contrôle de validité de l'information transmise.

Figure 2.1

Principe de calcul du code de contrôle de validité.



La correction des erreurs se fait soit par l'intermédiaire d'une nouvelle tentative de transmission, soit en exploitant la richesse des informations de redondance, qui localisent et

corrigent les erreurs détectées. Lorsqu'on utilise une simple détection d'erreurs, le récepteur n'a aucun moyen de localiser l'erreur détectée : celle-ci peut se situer aussi bien dans le champ de redondance que dans le champ des données. Dans tous les cas, la trame est considérée comme invalide et ignorée du récepteur. Il est toujours possible que des erreurs de transmission apparaissent et que, par malchance, la cohérence reste vraie. On se trouve alors en présence d'*erreurs résiduelles*. Dans ce cas, le mécanisme de contrôle d'erreurs n'a pas pu détecter que plusieurs erreurs de transmission se sont mutuellement compensées. Le taux d'erreurs résiduelles doit être aussi faible que possible mais il ne peut jamais être nul sur une liaison de données réelle.

Remarque

Les erreurs résiduelles peuvent également exister sur des liaisons de données employant des codes correcteurs d'erreurs. Le nombre d'erreurs compensées a alors dépassé les capacités de correction du code correcteur, et le protocole de liaison ne peut pas signaler cet événement.

La détection des erreurs résiduelles ne s'effectue donc pas au niveau du protocole de liaison mais à des niveaux plus élevés de l'architecture de communication : par exemple, au niveau de l'application ou même seulement par l'utilisateur qui a reçu une information incohérente par rapport à ce qu'il attendait.

Contrôle de la validité : protection au niveau du code

La protection au niveau du code consiste à organiser une redondance interne à celui-ci : parmi toutes les combinaisons possibles, certaines sont retenues comme valides. Ce type de protection est possible lorsque l'émission des données se fait par caractère (on introduit une redondance pour chaque caractère transmis). Par exemple, on ajoute à chaque caractère un *bit de parité* dit *parité verticale* ou VRC (*Vertical Redundancy Check*)[1], calculé comme suit : pour chaque caractère, on fait la somme modulo 2 de ses bits. Si le nombre de bits 1 est pair, on ajoute 0 à la fin du caractère, et si le nombre de bits 1 est impair, on ajoute 1.

Le contrôle de validité par VRC est fréquemment utilisé avec le code CCITT n° 5 sur les liaisons asynchrones. Par exemple, pour le caractère *M* codé par `1001101`, le bit de parité vaut 0. On transmet dans cet ordre `10110010` (les 7 bits de données en commençant par les poids faibles puis le bit de parité). L'inconvénient général lié aux contrôles par parité est qu'on ne détecte pas les erreurs doubles.

Contrôle de la validité : protection au niveau de la trame

La protection au niveau des trames consiste à rajouter une redondance à chaque trame, en fonction de l'ensemble des éléments binaires qui la constituent. Plusieurs techniques sont envisageables, mais nous n'examinerons ici que la parité longitudinale et le contrôle polynomial, qui sont les méthodes les plus connues et les plus utilisées.

Contrôle de parité longitudinale ou LRC (*Longitudinal Redundancy Check*) Pour améliorer la détection des erreurs dans les transmissions utilisant les contrôles par parité, on associe souvent parité longitudinale et parité verticale (VRC + LRC). Pour cela on ajoute, à la fin de la trame, un mot de code appelé *parité longitudinale* ou LRC, constitué par la somme modulo 2 de tous les bits de même rang.

1. Par référence aux bandes magnétiques utilisées comme mémoire de masse. Sur la bande, vue comme un long ruban, on écrivait, d'une part, en parallèle les bits du caractère, d'où le nom de par té verticale donné à cette méthode de protection du caractère, et, d'autre part, les caractères les uns à la suite des autres, dans le sens longitudinal de la bande, pour constituer le bloc de données. Un autre code de protection était inscrit en fin de bloc.



Exemple

Soit la suite de caractères *L*, *2*, *M* à transmettre, codée en CCITT n° 5 par les valeurs hexadécimales *4C*, *32* et *4D*. En parité paire, les bits de parité (en gras dans le texte) pour chaque caractère valent respectivement 1, 1 et 0. Le caractère de parité longitudinale est calculé comme suit :

`1 1 0 0 1 1 0 0` caractère L + parité VRC ;

`1 0 1 1 0 0 1 0` caractère 2 + parité VRC ;

`0 1 0 0 1 1 0 1` caractère M + parité VRC ;

`0 0 1 1 0 0 1 1` caractère du LRC à ajouter à la fin du bloc de données comme caractère de contrôle.

La suite des éléments binaires émise est donc `0011 0011 0100 1101 1011 0010 1100 1100`, si on transmet les caractères les uns derrière les autres, en commençant par les poids faibles de chaque caractère.

Contrôle polynomial Le contrôle polynomial, appelé couramment par abus de langage *code cyclique* ou CRC (*Cyclic Redundancy Check*), est très utilisé dans les protocoles modernes car il permet de détecter les erreurs sur plusieurs bits. Nous nous contentons ici d'en décrire le processus sans en faire la théorie.

Dans le contrôle polynomial, on considère la trame à transmettre comme un groupe de bits auquel on fait correspondre un polynôme $P(x)$, tel que le coefficient de degré i correspond à la valeur du i^e bit. Les algorithmes de calcul se font modulo 2 sur les polynômes [par exemple, $(x^7 + x^3) + (x^3 + x) = x^7 + x$]. On choisit un polynôme $G(x)$ de degré r, appelé *polynôme générateur*, caractéristique du contrôle. À l'émission, on multiplie $P(x)$ par x^r et on divise le polynôme obtenu par $G(x)$. Le reste noté $R(x)$, obtenu par division euclidienne, est de degré strictement inférieur à r. Il est ajouté à la fin de la trame comme code de contrôle. Ainsi :

$$x^r * P(x) = G(x) * Q(x) + R(x). \quad (1)$$

On transmet le polynôme $T(x)$, constitué à partir de $P(x)$ et du reste $R(x)$ et défini par l'équation (2) :

$$T(x) = x^r * P(x) + R(x). \quad (2)$$

D'après les équations (1) et (2) et en tenant compte du fait que les calculs s'effectuent modulo 2, ce polynôme vérifie :

$$T(x) = G(x) * Q(x).$$

Il est donc divisible par G(x).

Nous avons vu que le circuit de données peut modifier l'information. Soit $E(x)$ le polynôme associé aux erreurs apportées par le circuit. Les données reçues ont pour polynôme associé $S(x)$, défini par : $S(x) = T(x) + E(x)$. À la réception, on divise $S(x)$ par $G(x)$ et on obtient un reste $R_1(x)$ qui vérifie l'équation suivante :

$$S(x) = G(x) * Q_1(x) + R_1(x).$$

Si $R_1(x)$ est nul, on considère que $E(x)$ est nul et que l'information reçue correspond à celle émise. Si $R_1(x)$ n'est pas nul, le polynôme $E(x)$ ne l'est pas non plus : le circuit de données a introduit une ou plusieurs erreurs et l'information reçue doit être ignorée.

À l'information `1000001110000100` est associée $P(x) = x^{15} + x^9 + x^8 + x^7 + x^2$.
Soit le polynôme générateur de degré 12 :

$$G(x) = x^{12} + x^{11} + x^3 + x^2 + x + 1.$$

La division de $x^{12} * P(x)$ par $G(x)$ donne :

$$R(x) = x^{11} + x^9 + x^8 + x^7 + x^6 + x^4 + 1.$$

On transmet :

```
1 0 0 0 0 0 1 1 1 0 0 0 0 1 0 0          1 0 1 1 1 1 0 1 0 0 0 1
```

$x^{12}*P(x)$ $R(x)$

À la réception, on vérifie que le reste de la division par *G(x)* est nul.

1.3 Modes d'exploitation d'une liaison de données

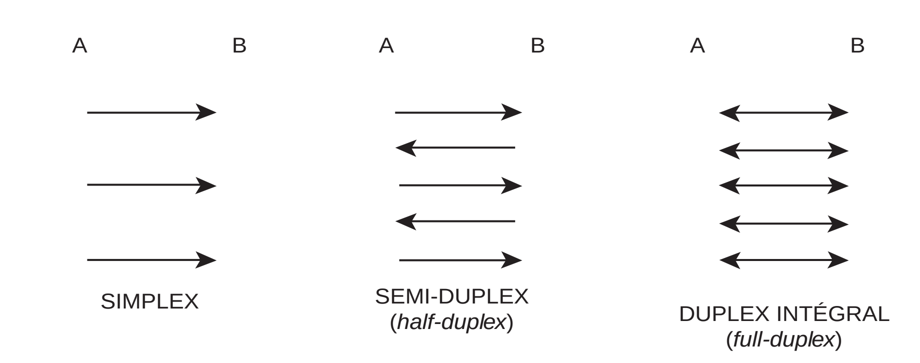
Le mode d'exploitation d'une liaison de données dépend du choix fait par le protocole de liaison (voir figure 2.2). La liaison peut être exploitée en *simplex*, *semi-duplex* (ou *half-duplex* ou *à l'alternat*), *duplex intégral* (ou *full-duplex*). Dans le mode simplex, l'échange de données se fait dans un seul sens. En semi-duplex, il se fait alternativement dans les deux sens, c'est-à-dire que les deux stations ne doivent pas transmettre simultanément (sinon il y a *contention*). En duplex intégral, les stations peuvent émettre simultanément sans aucune contrainte.

Remarque

Le mode d'exploitation de la liaison de données peut différer des caractéristiques du circuit de données : par exemple, on peut exploiter une liaison en semi-duplex alors que le circuit de données autorise des transmissions en duplex intégral.

Figure 2.2

Modes d'exploitation de la liaison.



2 Fonctionnalités d'un protocole de liaison

À partir d'un exemple simple de dialogue entre deux équipements, nous introduisons les différents types de protocoles de liaisons de données et les concepts qu'ils utilisent.

2.1 Représentation des échanges de données

Pour représenter les échanges de données, considérons que le temps s'écoule selon un axe horizontal. La transmission d'une trame est schématisée par un trait gras à la figure 2.3 et sa longueur représente la durée d'émission de la trame. Une flèche légèrement inclinée par rapport à la verticale représente la propagation de la trame ; la fin de la seconde flèche (côté destinataire) représente l'instant où la trame est totalement reçue.



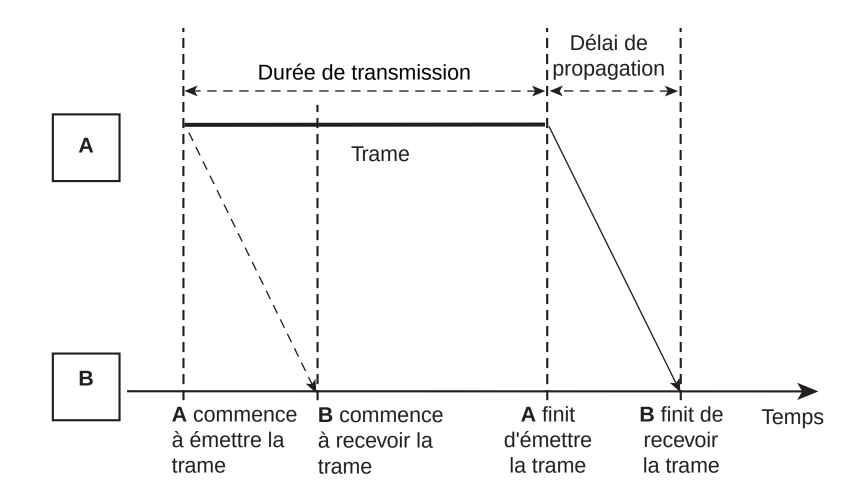


Figure 2.3

Représentation des échanges.

2.2 Contrôle de flux

Considérons deux équipements *A* et *B* reliés par un circuit de données sur lequel *A* veut envoyer des données à *B*. L'équipement *A* découpe les données en trames, appelées trames d'*information*, et les transmet les unes à la suite des autres. Elles sont repérées par la lettre *I*. Le circuit étant exempt d'anomalies, toutes les données sont délivrées sans erreur à l'équipement *B* qui les stocke pour les exploiter.

Supposons que *A* soit un ordinateur et *B* une imprimante lente, dotée d'une capacité mémoire limitée, lui imposant de garder en mémoire toutes les informations envoyées par *A* tant qu'elles ne sont pas imprimées. Si le rythme d'envoi des informations est nettement supérieur à son rythme d'impression, il y a rapidement saturation de la mémoire et perte d'informations par *B*. Il faut mettre en place un mécanisme de contrôle du rythme d'envoi des informations vers le récepteur, appelé *contrôle de flux*. Le petit engrenage de la figure 2.4 représente le mécanisme mis en jeu pour contrôler le flux des informations arrivant dans l'imprimante.

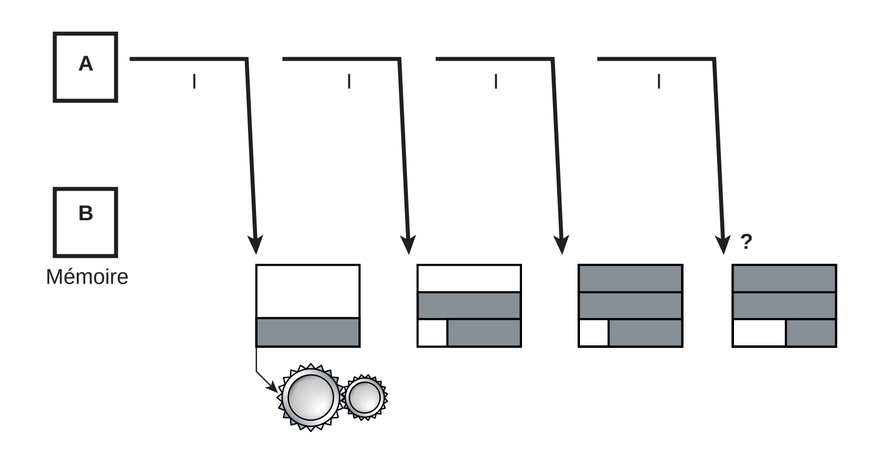


Figure 2.4

Exemple d'une transmission sans contrôle de flux sur un circuit de données parfait.

Pour réaliser le contrôle de flux, on introduit deux trames de *supervision*, RR (*Receiver Ready*) et RNR (*Receiver Not Ready*). Ces trames ne transportent aucune information utile et ne servent qu'à la gestion du dialogue. Elles sont générées et exploitées par le protocole de liaison et sont invisibles pour l'utilisateur. Le mécanisme est le suivant : à chaque réception de trame, l'équipement *B* envoie une trame RR s'il est prêt à accepter d'autres trames ou une

trame RNR s'il ne veut plus en recevoir de nouvelles. Dans ce dernier cas, *B* envoie RR dès qu'il est prêt à accepter de nouvelles trames, comme le montre la figure 2.5.

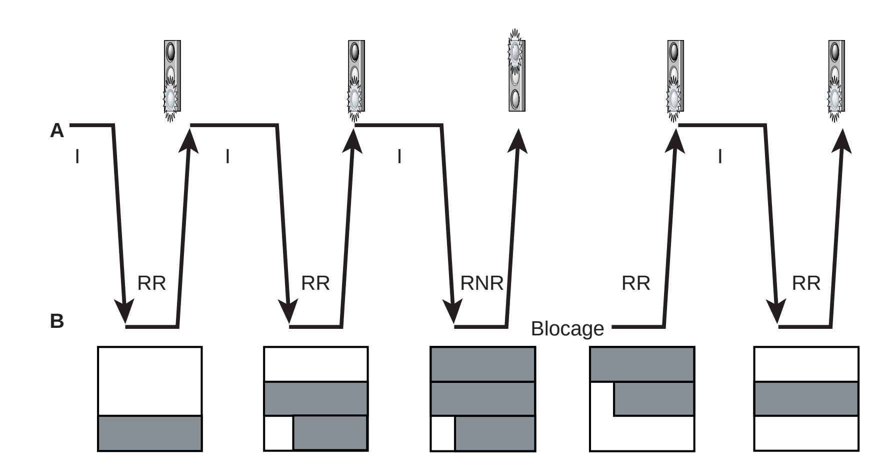


Figure 2.5

Mécanisme du contrôle de flux.

Il existe des variantes à ce mécanisme : par exemple, l'équipement *B* peut s'abstenir d'envoyer des trames RR. Lorsque la mémoire disponible descend au-dessous d'un certain seuil, l'équipement génère une ou plusieurs trames RNR et envoie des trames RR dès qu'une partie de la mémoire est libérée. Dans une autre variante, *B* peut transmettre en continu des trames RR tant qu'il a de la mémoire disponible (quelle que soit l'action de l'équipement *A*) et des trames RNR dès que sa mémoire est pleine. Un tel processus est couramment utilisé pour les liaisons entre ordinateur personnel et imprimante. Les trames de supervision sont alors réduites à deux caractères : la trame RNR est codée par le caractère *XOFF* (Ctrl+S), la trame RR par *XON* (Ctrl+Q).

2.3 Gestion des acquittements

Supposons maintenant que le circuit ne soit pas totalement fiable et introduise des erreurs. Au mécanisme de contrôle de flux décrit précédemment, il faut ajouter un processus d'*acquittement des trames d'information reçues.* Plusieurs options sont possibles :

- Lorsque l'équipement récepteur reçoit correctement une trame, il envoie une trame d'acquittement et ne fait rien en cas de mauvaise réception.
- Lorsqu'une trame est mal reçue, l'équipement récepteur envoie une *demande de retransmission* à l'émetteur et ne fait rien en cas de bonne réception.

Dans la seconde stratégie, l'absence de réponse est considérée comme un acquittement : à chaque trame, l'équipement émetteur arme un *temporisateur* correspondant à l'attente maximale d'une demande de retransmission provenant du récepteur ; si une telle demande parvient à l'émetteur, il répète la dernière trame. Dans le cas contraire, à expiration de la temporisation, l'émetteur considère que la transmission s'est bien effectuée. Cette stratégie présente deux inconvénients : elle est peu fiable car la demande de retransmission elle-même peut être mal transmise. Elle est en outre peu efficace puisqu'elle provoque une attente systématique en cas de bonne transmission.

Dans les protocoles de liaison de données, on utilise plutôt une stratégie d'acquittement positif, à l'aide des trames de supervision précédentes (RR et RNR). La stratégie de fonctionnement de *B* devient :

- Si *B* reçoit une trame correcte, l'équipement envoie un acquittement (trame RR ou RNR), selon l'état de sa mémoire pour assurer le contrôle de flux.

- S'il reçoit une trame erronée, il ne la mémorise pas et ne renvoie rien, comme s'il n'avait rien reçu.

Le principe de gestion des trames est le suivant : à l'émission de chaque trame d'information, *A* arme un temporisateur (correspondant à l'attente maximale de l'acquittement de *B*) qui sera désarmé à réception de l'acquittement correspondant. À l'échéance de cette temporisation, si *B* n'a pas répondu ou si sa réponse est brouillée, *A* réémet la trame et réitère le processus précédent. Le nombre de répétitions autorisées est limité : au-delà d'un certain seuil, on considère qu'un incident grave s'est produit (rupture totale de liaison, panne de l'équipement *B*, panne d'un élément de transmission sur *A* ou *B*…). Il faut alors avertir l'utilisateur que la liaison de données est rompue.

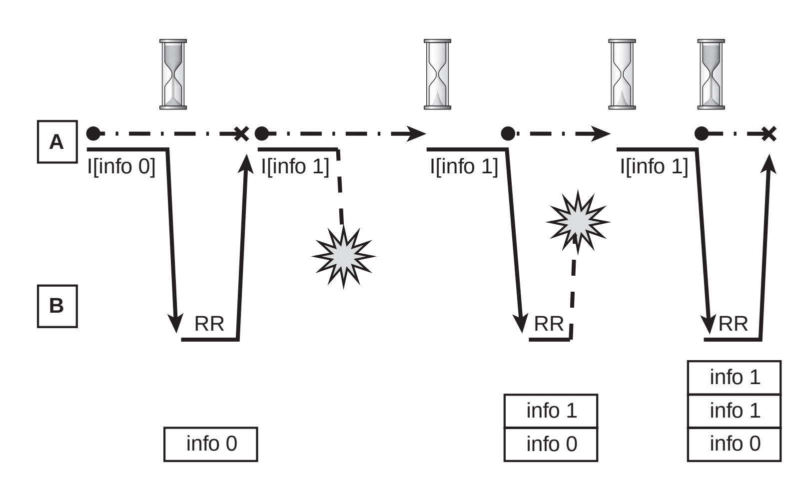
Ce mode d'acquittement n'est pas encore satisfaisant, car le circuit de transmission peut corrompre aussi bien les trames émises par *A* que celles émises par *B*. La figure 2.6 montre un échange de données mal conduit : une trame mal reçue par le récepteur se représente par un trait pointillé sans flèche ; les sabliers symbolisent les temporisations associées aux trames.

Figure 2.6

Protocole avec acquittement simple.

I[info 0] signifie que la trame I transporte l'information « info 0 ».

Dans cet exemple, la trame l'« info 1 » est dupliquée.



En effet, supposons que *A* envoie vers *B* une trame contenant l'information 1 et que *B* réponde à cette trame bien reçue par une trame RR qui est mal transmise (que l'équipement *A* ne reconnaît pas). Il réémet la même trame qui sera dupliquée dans *B*. Or, celui-ci n'a aucun moyen de détecter la duplication, puisqu'en aucun cas il n'analyse le contenu de la trame (*A* peut très bien décider de transférer deux fois de suite la même trame !). Puisque le protocole de liaison de données doit être *complètement indépendant du contenu* des trames transférées, il faut introduire un mécanisme supplémentaire qui distingue deux trames successives différentes. L'introduction d'un champ supplémentaire de *numérotation* ou d'*indication de retransmission* répond à ce besoin.

2.4 Numérotation des trames d'information

Le protocole numérote chaque trame d'information. La numérotation est placée dans l'en-tête, tout comme le type de la trame. Deux trames possédant des numéros différents sont considérées comme transportant des unités de données distinctes. Le protocole du récepteur exploite l'en-tête pour vérifier si la trame est correcte et en séquence. Dans l'affirmative, l'information contenue dans le champ de données est délivrée à l'utilisateur.

On appelle *N(S)* [*S* pour *send*] la variable donnant le numéro de la trame. Cette variable, codée sur quelques bits, est prise modulo *M*, un entier qui peut prendre les valeurs : 2 (le minimum pour distinguer deux trames successives différentes), 8 ou 128. L'introduction

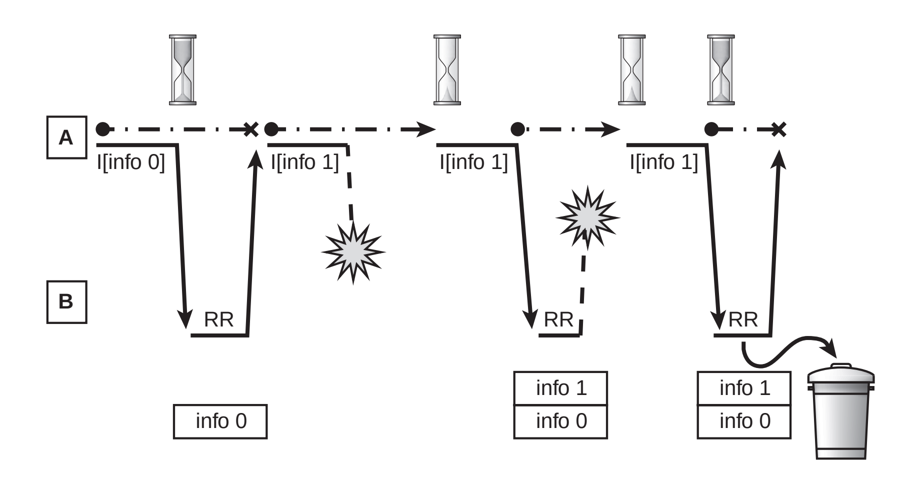
de numéros dans les trames impose des compteurs dans chaque station. De plus, il faut initialiser le dialogue pour que les deux stations se mettent d'accord sur les valeurs initiales des compteurs.

Le processus fonctionne de la façon suivante : *A* possède un compteur interne *V(S)* donnant le numéro de la prochaine trame à émettre. *A* émet cette trame en copiant *V(S)* dans le champ *N(S)*, puis il incrémente *V(S)*. Pour toute répétition d'une trame d'information, *A* émet la trame sans modifier son numéro d'ordre.

La figure 2.7 montre un échange de trames dont certaines sont erronées. La poubelle matérialise l'élimination des trames par le récepteur. Sur réception d'une trame, *B* teste la valeur du compteur *N(S)* : si sa valeur est égale à celle de la trame précédente, la trame est dupliquée et *B* l'ignore. Il peut cependant envoyer une trame de supervision pour acquitter la trame qu'il vient de recevoir.

Figure 2.7

Protocole à numérotation de trames.



Ce mécanisme possède une grosse faiblesse puisque la valeur de la temporisation est un paramètre critique. En effet, un mauvais choix de valeur peut entraîner un dysfonctionnement du protocole, comme le montre l'exemple de la figure 2.8.

Considérons le scénario où *A* envoie une trame I_0 bien reçue par *B* mais dont l'acquittement lui arrive après expiration de la temporisation associée ; *A* émet à nouveau la trame I_0. Lorsqu'il reçoit l'acquittement de la première trame I_0 émise, il considère qu'il s'agit de l'acquittement de la seconde et il émet la trame I_1. *A* interprète l'acquittement qui suit comme l'acquittement de sa trame I_1. Si, par malheur, I_1 est mal transmise, elle va manquer à *B : A* croit qu'elle est acquittée alors qu'en réalité *B* ne l'a pas reçue. On est donc obligé de préciser dans une trame d'acquittement le numéro de la trame d'information qu'elle acquitte.

2.5 Notion de fenêtre

Afin d'augmenter l'efficacité du dialogue, on introduit la notion d'*anticipation,* c'est-à-dire la possibilité d'émettre plusieurs trames à la suite, sans avoir reçu l'acquittement des trames précédentes. Ainsi, une trame de supervision n'acquitte plus une seule trame mais *un ensemble de trames qui se suivent sans erreur*. Le nombre de trames successives qu'on peut émettre sans réception d'acquittement est limité par une valeur notée *k*, appelée *fenêtre*. Considérons une numérotation des trames modulo 8. Intuitivement, on perçoit que la valeur maximale de *k* est au plus égale à 8. Nous montrons qu'elle doit être au plus de 7.



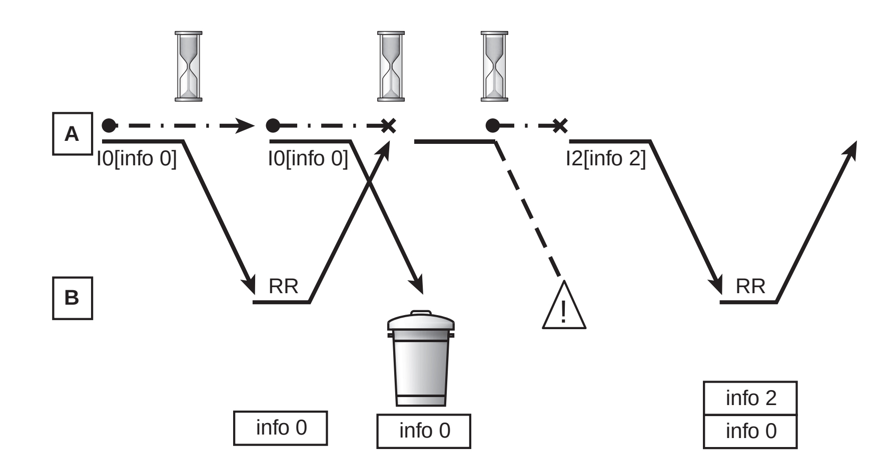


Figure 2.8

Exemple de mauvais fonctionnement avec des trames numérotées et des acquittements sans numérotation.

Adoptons une représentation circulaire des trames en attente d'acquittement. À la figure 2.9, un disque représente l'ensemble des valeurs des numéros d'une trame ; chaque trame occupe une part du disque. Un trait gras représente la valeur *V(S)* du compteur interne de *A* donnant le numéro de la prochaine trame à émettre.

À l'instant initial, *A* est au repos. Dès que *A* lance l'émission de la trame 0, cette trame est considérée comme émise mais non acquittée : on noircit la portion 0 du disque. Si *A* émet plusieurs trames successives, on noircit l'ensemble des trames en attente d'acquittement. Lorsque *A* reçoit un acquittement, les portions correspondant aux trames acquittées sont blanchies. L'ensemble des portions noircies représente l'état de la *fenêtre d'émission*. On représente également par un disque l'état du récepteur en noircissant les numéros que *B* s'attend à recevoir. À l'initialisation, *B* s'attend à recevoir la trame numérotée 0 : la case 0 est noircie. Lorsque cette trame est reçue correctement, *B* se met en attente de la trame 1 ; la case 1 est, par conséquent, noircie à son tour et ainsi de suite.

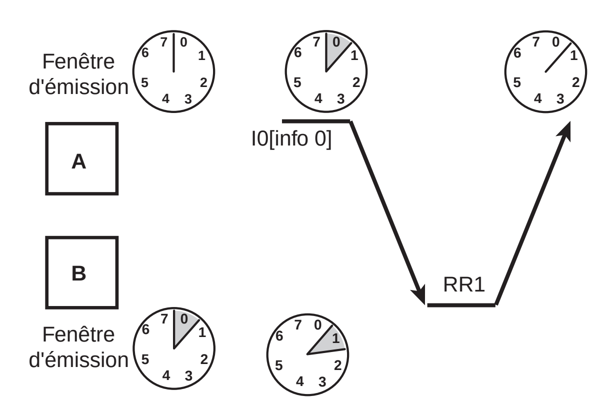


Figure 2.9

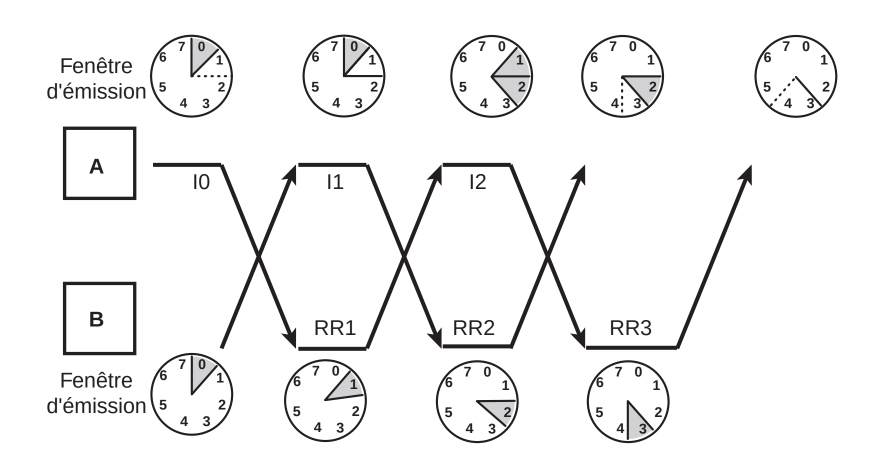
Représentation des fenêtres d'émission et de réception.

En absence d'erreur de transmission, le fonctionnement est le suivant : dès que *B* reçoit une trame, il enregistre son numéro *N(S)*, l'incrémente de 1, le mémorise dans une variable interne *V(R)* puis place cette valeur dans le champ *N(R)* de la trame de supervision qu'il renvoie à *A*. Tant qu'il en reste, l'équipement *A* émet ses trames, à moins qu'il n'ait atteint le nombre de trames autorisées sans réception d'acquittement. Une temporisation, armée à l'émission de chaque trame, est désarmée chaque fois que la trame correspondante est acquittée. On remarque que ce protocole suppose des équipements fonctionnant en mode duplex intégral car les acquittements sont reçus pendant l'émission des trames. La figure 2.10 illustre ce procédé.

Figure 2.10

Scénario pour un protocole à fenêtre d'anticipation de largeur 2.

Lorsque A a émis les trames I_0 et I_1, sa fenêtre est fermée. Il attend la réception d'un acquittement pour pouvoir émettre la trame I_2.



Déterminons par un exemple la taille maximale de la fenêtre d'anticipation, lorsque les trames sont numérotées sur 3 bits (les valeurs vont de 0 à 7). Considérons les deux scénarios suivants :

- *A* transmet une trame numérotée 0, acquittée par *B*. L'acquittement n'est pas reçu par *A* qui émet à nouveau la trame 0 à expiration de la temporisation associée.
- *A* transmet une trame numérotée 0, acquittée par *B* à l'aide de la trame *RR1*. Ensuite, huit trames sont successivement émises (les trames 1 à 7 puis la trame 0), mais les sept premières ne sont pas reçues par *B*.

Dans le dernier scénario, de son point de vue, *B* a reçu deux trames successives portant l'indice 0, autrement dit, deux fois la même trame alors qu'il s'agit de deux trames différentes. Il est donc nécessaire de limiter la fenêtre d'anticipation à sept trames pour éviter toute confusion. De façon générale, si les trames sont numérotées de 0 à *n*, la taille maximale de la fenêtre d'anticipation est au plus *n* : graphiquement, il doit toujours y avoir une part du disque non noircie pour éviter toute ambiguïté dans l'acquittement.

Quand une trame reçue est erronée, elle n'est pas prise en compte. Une erreur ne sera alors détectée que si l'une des trames I suivantes est correctement reçue. Par contre, son numéro ne correspondra pas au *N(S)* attendu. Deux stratégies sont envisageables : on réémet toutes les trames à partir de la trame erronée (*Go-back-N*) ou on ne réémet que la trame erronée par un mécanisme de *rejet sélectif* (*Selective Reject*).

2.6 Protocole Go-back-N

Dans la stratégie Go-back-N (retour au *n-ième*), une trame de supervision appelée REJ (*Reject*) sollicite la retransmission des trames à partir de la trame erronée.

Le Go-back-N est illustré à la figure 2.11 et respecte le scénario suivant : *A* envoie la trame 0 (mal reçue), suivie de la trame 1 (bien reçue). En recevant la trame 1, *B* constate une rupture de séquence : il a reçu la trame 1 sans avoir reçu de trame 0. De ce fait, il ne mémorise pas la trame 1 et envoie une trame REJ avec le numéro 0 pour demander la reprise d'émission à partir de la trame 0. En recevant la trame REJ0, *A* interrompt éventuellement l'émission de la trame en cours pour reprendre le processus d'émission à partir de la trame erronée.



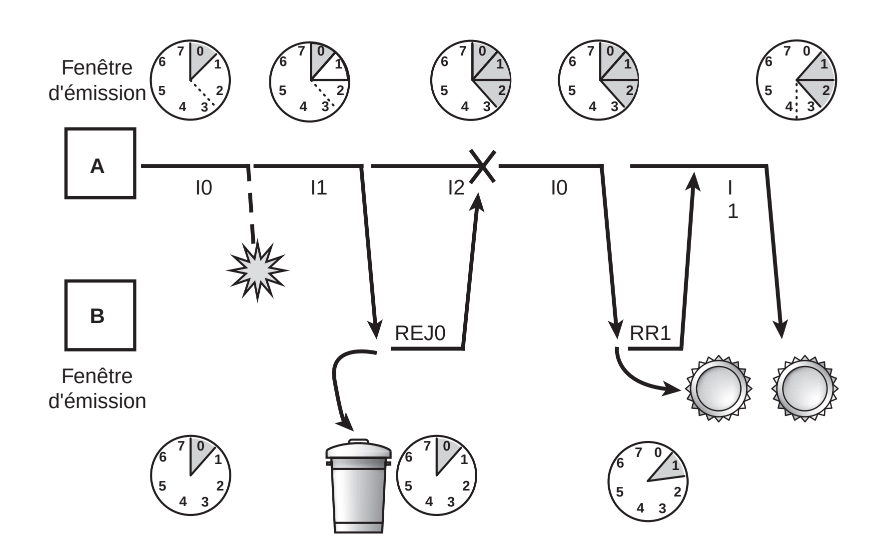


Figure 2.11

Scénario d'un protocole Go-back-N.

2.7 Piggy-backing

Dans des échanges bidirectionnels, chaque équipement envoie des trames d'information numérotées et acquitte les trames I qu'il a reçues (voir figure 2.12). Il y a donc deux sortes de trames émises : les trames I et les trames d'acquittement (RR, RNR ou REJ selon l'état de la réception). Un tel mécanisme n'est pas très efficace ; on l'améliore en utilisant les trames I pour véhiculer à la fois les informations à émettre et les acquittements des trames reçues précédemment. Chaque trame I possède deux numéros : un numéro *N(S)* [le numéro d'ordre de la trame I], et un numéro *N(R)* acquittant les trames émises dans le sens opposé. Ce mécanisme est appelé *piggy-backing*. Enfin, lorsqu'une station n'a pas de trame I à émettre, elle peut toujours utiliser des trames RR pour acquitter le trafic qu'elle reçoit.

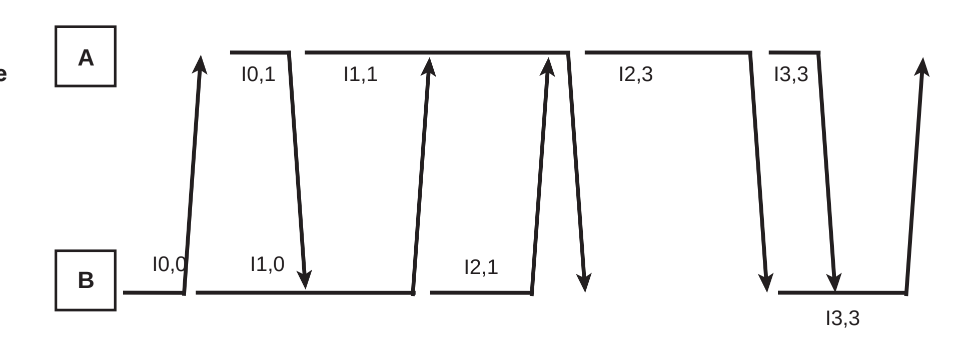


Figure 2.12

Scénario d'un protocole duplex intégral avec piggy-backing.

Une trame I N(S),N(R) a comme signification I N(S) et RRN(R).

La fenêtre d'anticipation à l'émission est de taille 3 au moins dans cet exemple.

2.8 Conclusions

Un protocole de liaison de données peut offrir plusieurs services suivant la qualité de la transmission :

- *Service sans acquittement, ni connexion, ni contrôle de flux* lorsqu'on souhaite utiliser un protocole très simple ou lorsque le circuit de données est d'excellente qualité.
- *Service avec acquittement mais sans connexion, ni contrôle de flux* qui permet d'améliorer un peu la fiabilité de la liaison mais ne garantit pas la non-duplication des messages.
- *Service avec acquittement, connexion et contrôle de flux* qui inclut la numérotation des trames et des acquittements. Ce service, le seul à offrir une réelle garantie de fiabilité,

est aussi le plus complexe à implanter. On distingue plusieurs stratégies dans la gestion des acquittements : le *Stop-and-Wait* (utilisation d'une fenêtre d'anticipation égale à 1), le *Go-back-N* et le *Selective Reject*. Le *Stop-and-Wait* est peu efficace, le *Go-back-N* est le plus utilisé ; le *Selective Reject* n'apporte pas de gain flagrant de performances dans la majorité des cas.

De multiples protocoles de liaison de données ont été développés. Nous nous contentons ici de la présentation détaillée d'un seul protocole : HDLC (*High level Data Link Control*), une recommandation internationale datant des années 1970.

3 Description du protocole HDLC (*High level Data Link Control*)

HDLC est le protocole normalisé par l'ITU (*International Telecommunications Union*[2]), qui décrit une transmission en duplex intégral fonctionnant sur une liaison point à point ; la transmission est synchrone et orientée bit. Ce protocole met en œuvre le mécanisme de transparence décrit en début de chapitre, ce qui le rend totalement indépendant du codage des données transportées. HDLC peut transporter des informations utilisant des codes de longueur variable. Sa variante la plus connue est de type Go-back-N avec un mécanisme de contrôle de flux. Il fonctionne en mode équilibré ou symétrique[3], c'est-à-dire que les deux stations ont les mêmes prérogatives et peuvent éventuellement fonctionner selon un mode half-duplex.

3.1 Structure d'une trame HDLC

La trame est la structure unique de longueur quelconque qui transporte toutes les informations. Un fanion en marque le début et la fin ; un seul fanion marque la fin d'une trame et le début de la suivante lorsque deux trames sont émises consécutivement. Le tableau 2.1 décrit les différents champs de la trame, dans leur ordre d'apparition :

- Le champ *Address* s'étend sur un octet et identifie une des extrémités de la liaison.
- Le champ *Control* décrit le type de la trame : il s'étend sur 1 octet (sur 2 octets dans le *mode étendu*).
- Le champ *Information* est facultatif. Il contient un nombre quelconque d'éléments binaires représentant les données de l'utilisateur.
- Le champ FCS (*Frame Control Sequence*) est la séquence de contrôle de trame, obtenue par un contrôle polynomial dont le polynôme générateur vaut $x^{16} + x^{12} + x^5 + 1$. Ce polynôme générateur est celui préconisé par la recommandation *V41* de l'ITU.

Tableau 2.1

Format de base des trames HDLC

Flag	Address	Control	Information	FCS	Flag
01111110	8 bits	8 bits	*N* bits	16 bits	01111110

Le champ de gauche est le premier transmis, le champ de droite est le dernier.

2. On trouve également le sigle français UIT (*Union internationale des télécommunications*) pour désigner la même instance internationale. Nous utiliserons le sigle anglais.

3. Dans ce mode, chaque équipement possède deux fonctions : une primaire, qui émet des requêtes, et une secondaire, qui envoie des réponses aux requêtes.



On commence par émettre les bits de poids faibles (du bit 1 au bit 8 de chaque champ). La transmission d'éléments binaires est continue ; en l'absence d'émission spécifique, les équipements émettent des suites de fanions pour maintenir la synchronisation entre les deux extrémités de la liaison de données.

3.2 DIFFÉRENTS TYPES DE TRAMES HDLC

Il existe trois types de trames identifiés par le champ Control : les trames d'information ou trames *I* permettent la transmission de données de l'utilisateur. Les trames de supervision ou trames *S* permettent l'acquittement et le contrôle de flux ; elles ne transportent pas de données, de même que les trames non numérotées ou trames *U* (*Unnumbered*)[4]. Ces dernières servent à commander la liaison : initialisation, libération, notification d'erreurs irrécupérables… Seule une trame *I* peut transmettre des données ; elle est numérotée par la variable *N(S)* et contient également l'acquittement des trames reçues en sens inverse (procédé de piggy-backing), grâce au numéro *N(R)*. Le tableau 2.2 montre le format du champ Control d'une trame I.

Tableau 2.2

Format de l'octet Control des trames I

8	7	6	5	4	3	2	1	
N(R)			P/F	*N*(S)			**0**	Format général des trames *I*

Le bit 1 de valeur 0 est spécifique à la trame I. La valeur du bit P/F dépend du statut de l'équipement (primaire ou secondaire) et de la nature de la trame (requête ou réponse[5]).

Trames de supervision (trames *S*)

Les trames *S* acquittent les trames *I* et indiquent l'état de disponibilité des stations (aptitude ou non à recevoir de nouvelles trames *I*). Contenant un numéro *N(R)*, elles servent au contrôle d'erreurs et au contrôle de flux. Les trois trames de supervision[6] sont :

- La trame RR indique que l'équipement est prêt à recevoir de nouvelles trames *I*. Le numéro *N(R)* donne le numéro de la prochaine trame attendue. Il signifie que *toutes* les trames *I* de numéro *N(S)* strictement inférieur à *N(R)* ont été reçues. Un équipement peut aussi envoyer des trames *RR* pour indiquer son état ou pour demander l'état de la station située à l'autre extrémité.
- La trame RNR acquitte les trames reçues et indique en outre que l'équipement n'est pas en mesure de recevoir de nouvelles trames *I*. Le numéro *N(R)* a la même signification que dans la trame *RR*.
- La trame REJ sert à demander l'arrêt immédiat des émissions en cours et une retransmission à partir de la trame *I* portant le numéro indiqué dans *N(R)*.

Le tableau 2.3 donne le format de l'octet Control des trames *S*.

Tableau 2.3

Format de l'octet Control pour les trames S de supervision

8	7	6	5	4	3	2	1	Format général des trames *S*
N(R)			P/F	0	0	**0**	**1**	*RR*
N(R)			P/F	0	1	**0**	**1**	*RNR*
N(R)			P/F	1	0	**0**	**1**	*REJ*

4. Nous verrons plus loin dans le protocole PPP qu'il existe des trames U, appelées trames UI (*Unnumbered Information*), qui transportent des données de l'utilisateur.

5. L'interprétation du bit 5 (bit *P* ou *F*) dépend du statut de l'équipement. Une trame de requête est émise par la fonction primaire qui gère le bit *P*. Une trame de réponse, émise par la fonction secondaire de l'autre équipement, utilise le bit *F*.

6. Une quatrième trame de supervision, la trame *SREJ*, a été définie pour le rejet sélectif, mais ce mode de rejet des trames erronées ne fait pas partie de la version normalisée décrite ici. Nous verrons son fonctionnement au cours des exercices.

Trames non numérotées (trames *U*)

On utilise les trames *U* pour les fonctions supplémentaires de commande de la liaison. Citons les principales :

- SABM (*Set Asynchronous Balanced Mode*) pour initialiser le fonctionnement en mode équilibré.
- DISC (*DISConnect*) pour rompre logiquement la liaison entre les deux stations.
- UA (*Unnumbered Acknowledgement*) pour acquitter des commandes comme SABM ou DISC.
- FRMR (*FRaMe Reject*) pour rejeter une commande invalide (correcte du point de vue de la détection des erreurs mais incohérente par rapport à l'état du dialogue).
- DM (*Disconnect Mode*) pour indiquer l'état de déconnexion d'une station. Elle s'utilise, en particulier, pour répondre négativement à une demande d'initialisation par SABM.

Le tableau 2.4 donne le format de l'octet Control des trames *U*.

Tableau 2.4

Format de l'octet Control pour les trames U

8	7	6	5	4	3	2	1	
0	0	1	P/F	1	1	**1**	**1**	*SABM*
0	1	0	P/F	0	0	**1**	**1**	*DISC*
0	1	1	P/F	0	0	**1**	**1**	*UA*
1	0	0	P/F	0	1	**1**	**1**	*FRMR*
0	0	0	P/F	1	1	**1**	**1**	*DM*

3.3 Établissement et libération de la liaison de données

HDLC est un protocole *orienté connexion* : il faut établir la liaison avant d'envoyer des informations ou en recevoir. De même, lorsque l'un des équipements veut terminer le dialogue, il engage une *procédure de libération*. La figure 2.13 décrit un problème survenu lors d'une tentative de connexion.

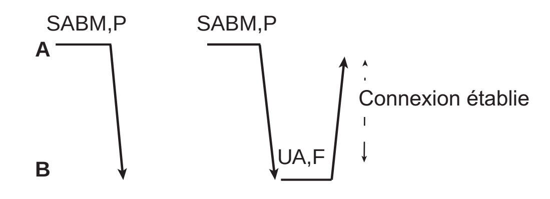


Figure 2.13

Initialisation de la liaison de données.

A tente d'établir une connexion vers B qui ne détecte pas la première trame mais acquitte la seconde.

N'importe quel équipement peut initialiser la liaison. Le primaire de l'équipement initiateur envoie la trame SABM,P et attend la réponse du secondaire de l'autre équipement. En cas de non-réponse, il réitère son envoi jusqu'au nombre maximal de tentatives de connexion. Au bout de ce nombre d'essais infructueux, il considère que la liaison est impossible à établir. À réception d'une SABM,P, le récepteur transmet une trame UA,F si son utilisateur accepte le dialogue, sinon il envoie une trame DM,F. Dans le premier cas, la connexion est alors établie ; tous les compteurs et les temporisateurs sont initialisés. Les premières trames émises de chaque côté porteront un *N(S)* égal à 0. Dans le second cas, les équipements entament une phase de libération. Ce dernier processus, symétrique à l'établissement de la liaison, utilise les commandes DISC et UA.

4 Cas particulier du protocole PPP (*Point to Point Protocol*)

Le protocole PPP est le protocole de liaison point à point utilisé dans Internet. Il utilise les lignes téléphoniques de l'abonné pour accéder au réseau (la liaison concerne typiquement un ordinateur personnel et le fournisseur d'accès à Internet). Il s'agit d'une version très simplifiée d'HDLC qui ne comprend – sauf options – ni contrôle de flux, ni mécanisme de reprise sur erreurs. La figure 2.14 donne la structure d'une trame PPP.



Figure 2.14

Format d'une trame PPP.

Les 8 bits du champ Address sont à 1 (la liaison étant point à point, une seule valeur d'adresse suffit).

Le champ Control a la même signification que dans HDLC.

Le champ Data PPP commence par deux octets (le champ protocole), qui identifient le protocole de niveau supérieur auquel est destinée la trame ; il se termine par un champ FCS dont le mode de calcul est identique à celui d'une trame HDLC.

La seule trame transportant des données sur une liaison fiable est une trame *U* de type *UI* (*Unnumbered Information*). Cette trame contient un champ d'informations mais n'est pas numérotée (car il n'y a pas de contrôle de flux). L'absence de mécanisme de reprise sur erreur ne signifie pas que le circuit est fiable : le champ FCS sert à valider les trames reçues.

PPP comprend également un ensemble de sous-protocoles choisis à l'ouverture de la liaison de données pour sécuriser les échanges : LCP (*Line Control Protocol*), PAP (*PPP Authentification Protocol*), CHAP (*Challenge Authentification Protocol*) et NCP (*Network Control Protocol*).

À l'initialisation d'un transfert, PPP négocie les paramètres de l'échange par le protocole LCP ; PAP autorise l'échange – en clair – des mots de passe avant le transfert des données. Si on souhaite un échange sécurisé, on peut utiliser CHAP, qui effectue un chiffrement tout au long de la communication grâce à un échange préalable de clés publiques et de clés secrètes (voir les compléments pédagogiques sur le site www.pearsoneducation.fr). Enfin, le protocole NCP sert à négocier les paramètres de connexion (les options de transfert choisies par chaque extrémité de liaison, indépendamment l'une de l'autre) et les paramètres de niveau réseau (par exemple, l'affectation des adresses IP, que nous verrons au chapitre 6).

Résumé

Le protocole de liaison de données supervise le circuit de données et définit un ensemble de règles pour assurer la fiabilité des échanges. Il spécifie le format des trames, les moyens de contrôler leur validité, ainsi que les règles du dialogue entre les deux extrémités de la liaison. Il exerce aussi un contrôle de flux pour maîtriser le rythme d'envoi des informations et valider la réception des informations reçues.

HDLC est un exemple de protocole normalisé très répandu, qui gère des trames de données à l'aide de trames de supervision. Il fonctionne en full-duplex et permet la reprise sur erreur. Il garantit en outre l'ordre des données. PPP, utilisé dans Internet, en est une version très simplifiée qui n'exerce pas de contrôle de flux mais propose plusieurs modes d'échanges de données, négociées avant tout transfert de données entre les deux extrémités de la liaison.

Problèmes et exercices

Exercice 1 Problème lié à l'insertion du bit de transparence

Énoncé

Soit la suite de données binaires située dans le champ d'information d'une trame HDLC : 0111101111110011111100011.

a Quelle est la suite réellement fournie au support de transmission (pour ces données seulement) ?

b Que se passe-t-il si le douzième bit de la suite réellement transmise a été mal reconnu du récepteur ?

Solution

a Pour garantir la transparence, la suite réellement émise est : 01111011111**0**0011111**0**10001 (nous avons indiqué en gras les bits de transparence).

b Par suite de l'erreur de transmission, la suite réellement transmise devient : 01111 01111110 011111**0**10001. Le récepteur reconnaît un fanion dans la zone grisée : il considère donc que la trame se termine à cet endroit ! L'erreur est détectée car la trame ne respecte pas le format d'une trame I (de même, ce qui est pris pour la trame suivante n'a pas d'adresse valide). En outre, les 16 bits précédant le faux fanion sont considérés comme les bits du FCS, qui a toute chance d'être faux. Enfin, un récepteur ignore toute trame comptant moins de cinq octets.

Exercice 2 Transparence aux données transmises

Énoncé

a Écrire la suite des bits réellement transmise pour une trame SABM émise par un équipement d'adresse *A* (03 en hexadécimal) vers un équipement d'adresse *B* (01 en hexadécimal). Le bit n° 5 (bit *P*) est mis à 1. On admettra que le FCS de cette trame vaut en binaire 1101011111111011.

b Par quelle trame répond l'équipement *B* ?

Solution

a Il faut transmettre dans l'ordre : le fanion de début, l'adresse de l'équipement *B*, l'octet Control (correspondant à la trame SABM,P), le FCS puis le fanion de fin. En représentant les données de gauche à droite et en indiquant en gras les 0 insérés pour la transparence, la suite réellement transmise est :

01111110	00000001	11111**0**100	110101111**0**1111	01111110
fanion	adresse *B*	octet *SABM*	*FCS*	fanion

b L'équipement *B* répond par une trame UA,F, c'est-à-dire par une trame U composée de : fanion, adresse *B*[7], FCS, fanion.

7. Une trame de commande (SABM) contient l'adresse de l'équipement auquel est destinée la trame. Dans la trame de réponse (UA), l'équipement place sa propre adresse. Les deux trames contiennent donc la même adresse.



EXERCICE 3 CALCUL DU VRC ET DU LRC

Énoncé

Calculez le VRC et le LRC du message *HELLO* en utilisant la parité paire, sachant que *H* est codé par `0001001`, *E* par `1010001`, *L* par `0011001` et *O* par `1111001`. Précisez l'ordre de transmission du message construit.

Solution

```
        VRC
H   0001001 0
E   1010001 1
L   0011001 1
L   0011001 1
O   1111001 1
---------------------------
LRC 0100001 0
```

Message transmis (bits de poids faible en premier) :

LRC + O + L + L + E + H →

`01000010   11110011   00110011   00110011   10100011   00010010`

EXERCICE 4 DÉTECTION D'ERREUR PAR VRC ET LRC

Énoncé

On désire transmettre la suite de 16 bits de données : 2BE3 (en hexadécimal), le premier bit transmis correspondant au bit de poids faible du chiffre 2. La protection contre les erreurs se fait par parité verticale (de parité paire) et longitudinale.

a Donnez la suite de bits des quatre caractères et la suite binaire complète transmise au récepteur pour ce bloc de données.

b En supposant que, par suite d'une erreur de transmission, le 19e bit de la suite trouvée à la question a soit modifié, calculer la valeur du reste trouvée par le récepteur.

Solution

a Il faut ajouter, à chaque caractère, le VRC qui lui correspond puis calculer le LRC du bloc de données. Les résultats sont récapitulés au tableau 2.5 :

Tableau 2.5

VRC et LRC de la question a

Chiffre	Codage	VRC
2	0 0 1 0	1
B	1 0 1 1	1
E	1 1 1 0	1
3	0 0 1 1	0
LRC	0 1 0 0	1

Exercices

On envoie : LRC 3 E B 2 soit dans l'ordre d'émission :
01001 00110 11101 10111 00101.

b Le bit erroné et le LRC trouvés sont indiqués en gras au tableau 2.6 :

Tableau 2.6

Corrigé de la question b

Chiffre	Codage	VRC
2	0 0 1 0	1
B	1 0 1 1	1
E	1 1 1 0	1
3	0 **1** 1 1	0
LRC	**0 0 0 0**	**1**

EXERCICE 5 VRC, LRC ET CONTRÔLE POLYNOMIAL

Énoncé

On désire vérifier le bloc de données constitué par les deux octets codés avec une parité paire : 00110011 et 11110011.

a Quel est le LRC correspondant à ce bloc de données ?

b Représentez le LRC sous forme polynomiale.

c On désire vérifier ce bloc de données par un contrôle polynomial de polynôme générateur $x^8 + 1$. Donnez la valeur du polynôme LRC(x). Que constatez-vous ?

Solution

a Le calcul du LRC est donné tableau 2.7.

Tableau 2.7

LRC de la question a

octet 1	00110011
octet 2	11110011
LRC	11000000

b La forme polynomiale du LRC est : $LRC(x) = x^7 + x^6$.

c Le polynôme $M(x)$ du message est égal à : $x^{13} + x^{12} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x + 1$. Il faut diviser le polynôme $P(x) = x^8 * M(x)$ par $x^8 + 1$, c'est-à-dire :

$$x^{21} + x^{20} + x^{17} + x^{16} + x^{15} + x^{14} + x^{13} + x^{12} + x^9 + x^8 / x^8 + 1 = x^7 + x^6.$$

Les deux méthodes de calcul donnent le même résultat.

Remarque

Nous voyons que le LRC est, en fait, un contrôle polynomial de polynôme générateur $x^8 + 1$. Ce résultat est intéressant car il permet de calculer le LRC *à la volée*, au fur et à mesure de l'arrivée des bits en série, alors que, normalement, il se calcule en parallèle sur tous les bits des caractères composant le bloc de données.



Exercice 6 Calcul d'un contrôle polynomial

Énoncé Soit la suite d'éléments binaires `0 1 1 1 0 1 0 0 0 0 1 0 1 0 1`.

a Calculer le bloc de contrôle d'erreur pour ces données, en supposant qu'on utilise un code polynomial de polynôme générateur $x^5 + x^3 + 1$.

b On reçoit le bloc suivant : `0 0 0 1 0 1 0 0 0 0 1 0 1 0 1 0 0 0 1 0`. Le contrôle d'erreur utilisant le même polynôme générateur, quelle est la décision prise par le récepteur concernant ce bloc ?

Solution

a Le polynôme $M(x)$ correspondant au message est égal à $x^{13} + x^{12} + x^{11} + x^9 + x^4 + x^2 + 1$. Multiplions-le par x^5, ce qui donne :

$$P(x) = x^5 * M(x) = x^{18} + x^{17} + x^{16} + x^{14} + x^9 + x^7 + x^5.$$

Le reste $R(x)$ vaut $x^4 + x^2 + x + 1$. Le mot de code émis est :

$$P(x) = x^{18} + x^{17} + x^{16} + x^{14} + x^9 + x^7 + x^5 + x^4 + x^2 + x + 1.$$

b Le polynôme $M(x)$ correspondant au mot de code reçu vaut :

$$x^{16} + x^{14} + x^9 + x^7 + x^5 + x + 1.$$

Il n'est pas identique au mot de code émis. Effectivement, la division polynomiale donne un reste non nul, valant :

$$R(x) = x^4 + x^2 + 1.$$

Le récepteur refusera donc le bloc de données.

Remarque

Dans ce bloc de données, nous voyons que plusieurs bits ont été mal transmis, sinon le polynôme reçu serait identique à celui trouvé à la question a. Remarquons que les erreurs se trouvaient aussi bien dans le corps du message que dans le bloc de contrôle mais, cela, le récepteur ne pouvait pas le savoir !

Exercice 7 Détection d'erreur par contrôle polynomial

Énoncé On considère la suite de données binaires : `010110110000111101010100`, constituée de deux caractères de 8 bits suivis d'un bloc de contrôle d'erreur calculé à l'aide d'un code polynomial de polynôme générateur $x^8 + 1$. Le récepteur détecte-t-il des erreurs de transmission dans la suite reçue ? Pourquoi ?

Solution Soit $P(x) = x^{22} + x^{20} + x^{19} + x^{17} + x^{16} + x^{11} + x^{10} + x^9 + x^8 + x^6 + x^4 + x^2$ le polynôme représentant les deux caractères, suivi du reste de la division polynomiale. Le récepteur ne détecte pas d'erreur car le reste calculé par l'émetteur correspond à celui calculé par le récepteur. En effet, les deux octets de données valent : `01011011 00001111`, c'est-à-dire que $M(x) = x^{14} + x^{12} + x^{11} + x^9 + x^8 + x^3 + x^2 + x + 1$. En multipliant $M(x)$ par x^8, on retrouve les 16 premiers bits de $P(x)$. Si on effectuait la division polynomiale de $x^8 * M(x)$, on trouverait un reste $R(x) = x^6 + x^4 + x^2$. Le récepteur considère qu'il n'y a pas d'erreur.

Exercices

EXERCICE 8 CONTRÔLE POLYNOMIAL AVEC LE POLYNÔME V41

Énoncé Les premiers bits à transmettre dans une trame d'informations gérée par le protocole HDLC sont les suivants : `10101110`. Le polynôme générateur utilisé est le polynôme normalisé par la recommandation *V41*.

a Trouvez le reste de la division polynomiale du message par ce polynôme.

b En supposant que la transmission des 8 bits ait été effectuée sans erreur, représentez la suite des opérations effectuée par le récepteur depuis le premier bit de données jusqu'à la réception du dernier bit de contrôle.

Solution **a** $M(x) = x^7 + x^5 + x^3 + x^2 + x$.

La division polynomiale à effectuer est : $x^{16} * M(x)$, à diviser par $G(x)$, soit : $x^{23} + x^{21} + x^{19} + x^{18} + x^{17}$ à diviser par $x^{16} + x^{12} + x^5 + 1$.

b Le tableau 2.8 donne les restes et les quotients successifs calculés par le récepteur :

Tableau 2.8

Restes successifs du contrôle polynomial utilisant le polynôme V41

Restes successifs	Quotients successifs
$x^{23} + x^{21} + x^{19} + x^{18} + x^{17}$	–
$x^{21} + x^{18} + x^{17} + x^{12} + x^7$	x^7
$x^{18} + x^{12} + x^{10} + x^7 + x^5$	$x^7 + x^5$
$x^{14} + x^{12} + x^{10} + x^5 + x^2$	$x^7 + x^5 + x^2$

Ce qui donne un reste : $R(x) = x^{14} + x^{12} + x^{10} + x^5 + x^2$, soit en binaire sur 16 bits : `0101010000100100`.

EXERCICE 9 ÉCHANGE DE DONNÉES AVEC DES TEMPS DE PROPAGATION IMPORTANTS (CAS DES LIAISONS PAR SATELLITE)

Énoncé **a** Un équipement *A* dialogue avec un équipement *B* selon le protocole LAP-B, *via* une liaison satellite. Le satellite réémet les signaux reçus sans aucun traitement et se trouve à 200 km d'altitude. Tous les délais dus aux traitements sont négligeables. Les équipements dialoguent à 9 600 bit/s et limitent la taille du champ de données à 64 octets. Tout équipement recevant une trame d'information correcte l'acquitte immédiatement s'il n'a aucune information à transmettre. En faisant l'hypothèse qu'il n'y ait aucune erreur de transmission, déterminez quelle est la taille minimale de la fenêtre d'anticipation pour une transmission efficace, lorsque *A* est le seul équipement à émettre des données.

b Même question avec un satellite situé maintenant à 36 000 km d'altitude.

Solution **a** Soit T le temps de transmission, l la longueur en bits du message, t_p le temps de propagation (temps mis par le message émis à la vitesse de la lumière, c'est-à-dire à 300 000 000 m/s) et D le débit binaire en bit/s. Nous avons la relation : $T = l/D$.



A transmet un message à B, qui le reçoit à $T + t_p$ et B envoie immédiatement la réponse. Pour éviter toute perte de temps, il faut que la taille de la fenêtre soit telle qu'elle corresponde à un temps d'émission supérieur à T, augmenté du temps d'attente de la réponse. Calculons le temps de transmission d'un message et le temps de propagation. Le temps de transmission d'un message à 64 octets de données vaut :

$$T = (48 + 64*8)/9\,600 = 58{,}33 \text{ ms.}$$

À 200 km d'altitude, le temps de propagation vaut :

$$t_p = 2*200\,000/300\,000\,000 = 1/750 = 0{,}001333333 = 1{,}333 \text{ ms.}$$

Si Δ est le temps d'attente de réception de la trame RR :

$$\Delta = 2*t_p + t_{RR} = 2\ t_p + (48/9\,600) = 2*1{,}33 + 5 = 7{,}66 \text{ ms.}$$

Pendant qu'on envoie le second message par anticipation, on recevra le RR. Il n'y aura donc jamais de problème de taille de fenêtre.

b À 36 000 km d'altitude, le temps de propagation devient : $t_p = 2*36\,000/300\,000 = 240$ ms. Δ vaut alors :

$$\Delta = 2*t_p + t_{RR} = 485 \text{ ms.}$$

Soit n le nombre de messages écoulés avant qu'on finisse de recevoir la réponse. Nous trouvons :

$n = \Delta/T = 8{,}31$, ce qui est une valeur supérieure à la taille maximale de la fenêtre d'anticipation prévue dans le protocole HDLC. Il y aura un silence chaque fois que la fenêtre sera pleine. On pourrait améliorer le système si le modulo de la numérotation était plus grand ou si les messages envoyés étaient plus longs.

Exercice 10 Relation entre taille de fenêtre et modulo de la numérotation des trames

Énoncé

Dans un protocole de liaison de données, on suppose que chaque émetteur peut utiliser au maximum *Maxseq* + 1 numéros de séquence différents, comptés de 0 à *Maxseq*. Expliquer pourquoi la taille de la fenêtre en émission doit rester inférieure à *Maxseq*. Mettre en évidence un cas d'ambiguïté.

Solution

Soit W la taille de la fenêtre. Si elle est égale à la largeur N du champ de numérotation (correspondant au modulo du compteur), il y a confusion entre le message portant le numéro k et le message de numéro $k+N$, puisque les deux numéros ont la même congruence (ils ont le même reste modulo N).

Un cas d'ambiguïté est décrit dans ce qui suit :

Prenons une numérotation modulo 8. Les numéros possibles sont donc 0, 1, 2… 7 (*Maxseq* = 7). Si $W = 8 =$ *Maxseq* + 1, une station X qui émet plusieurs trames dont la première est mal transmise recevra un acquittement RR0. Si, maintenant, elle envoie huit trames consécutives avec succès, elle recevra également RR0 ! La station réceptrice va considérer que la trame 7 est un doublon de la trame 0 (puisque, pour elle, les deux trames portent le

même numéro). La station réceptrice va ignorer les huit trames qu'elle a pourtant reçues correctement…

Nous pouvons conclure de cet exemple que la taille maximale de la fenêtre doit être au plus égale à *Maxseq*.

EXERCICE 11 PREMIÈRE REPRÉSENTATION D'UN ÉCHANGE DE DONNÉES SELON LE PROTOCOLE HDLC

Énoncé

a On considère un échange de données bidirectionnel simultané entre deux stations *A* et *B*, géré par le protocole LAP-B (sous-ensemble de HDLC utilisant le rejet simple des erreurs). La liaison est déjà initialisée et aucun temporisateur n'est armé. La station *A* a 5 trames d'information à transmettre à la station *B*. Celle-ci en a 10 à émettre vers *A*. On suppose que toutes les trames sont de même longueur. Les deux stations commencent leur transmission à des instants très voisins l'un de l'autre. Donnez le schéma des échanges en indiquant le numéro des trames émises et reçues avec la nomenclature classique : *I, N(S), N(R)* pour une trame I portant le numéro *N(S)* et acquittant les trames jusqu'au numéro *N(R) – 1*. On suppose de même :

- Qu'il n'y a aucune erreur de transmission sur les trames de *A*.
- Qu'il y a une erreur de transmission sur les sixième et septième trames de *B*.
- Que les trames sont correctement retransmises.
- Que le temps de propagation est équivalent au quart de la durée de transmission d'une trame I.
- Que la taille de la fenêtre d'anticipation est maximale.
- Que les accusés de réception (dont la durée de transmission est égale au quart de la durée d'une trame I) sont transmis dès que possible. Ils sont en tout cas inclus dans des trames I s'il y en a.

b Que se passerait-il si les trames de *A* étaient dix fois plus longues que celles de *B* ?

Solution

a Le diagramme de la figure 2.15 décrit la suite des trames échangées par *A* et *B*.

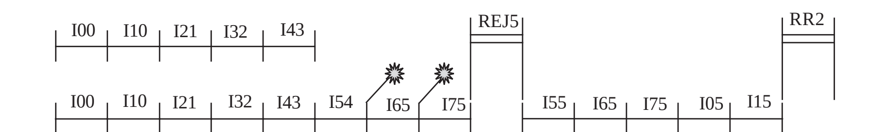


Figure 2.15
Diagramme d'échange de trames entre A et B.

Remarque

Il faut que *A* attende la huitième trame de *B* pour détecter la rupture de séquence. Cette trame a pu être émise car le nombre de trames émises par *B* mais non encore acquittées est inférieur à la taille de la fenêtre. Cela montre que la dernière trame I erronée d'une station ne peut pas être signalée par une trame REJ. Seule l'expiration de la temporisation d'attente d'acquittement de cette trame provoquera sa réémission.



b *B* aurait le temps d'envoyer toutes les trames de la fenêtre avant d'avoir reçu la première trame de *A*. Tout se passerait comme si les échanges se déroulaient à l'alternat (semi-duplex).

EXERCICE 12 REJET SIMPLE ET REJET SÉLECTIF DE TRAMES ERRONÉES (DEUXIÈME REPRÉSENTATION)

Énoncé

On considère deux stations *A* et *B* utilisant la version LAP-B du protocole HDLC. Dans cet exercice, nous supposons que la liaison de données est correctement initialisée et que le transfert de données peut démarrer dès que possible. La taille de la fenêtre d'anticipation est 2 ; l'échange de données est full-duplex et les deux stations démarrent simultanément le transfert de données. *B* n'a qu'une seule trame d'informations à émettre et sa trame est sans erreur. *A* a 3 trames d'informations à émettre. Nous allons envisager plusieurs scénarios pour le transfert de données de *A* :

a Dans le premier cas, la première trame de *A* est erronée la première fois qu'elle est émise mais elle est retransmise correctement. Donnez le diagramme correspondant aux différentes trames échangées entre *A* et *B*.

b Dans le deuxième cas, la deuxième trame de *A* est erronée la première fois qu'elle est émise mais elle est retransmise correctement. Donnez le diagramme correspondant aux différentes trames échangées entre *A* et *B*.

c Dans le dernier cas, les deux dernières trames de *A* sont erronées la première fois qu'elles sont émises mais elles sont retransmises correctement. Donnez le diagramme correspondant aux différentes trames échangées entre *A* et *B*.

On considère maintenant que les deux stations *A* et *B* utilisant la version du protocole HDLC demandant la réémission des trames erronées en utilisant le rejet sélectif SREJ.

d En reprenant les mêmes hypothèses de travail qu'à la question a, donnez le diagramme correspondant aux différentes trames échangées entre *A* et *B*.

e En reprenant les mêmes hypothèses de travail qu'à la question b, donnez le diagramme correspondant aux différentes trames échangées entre *A* et *B*.

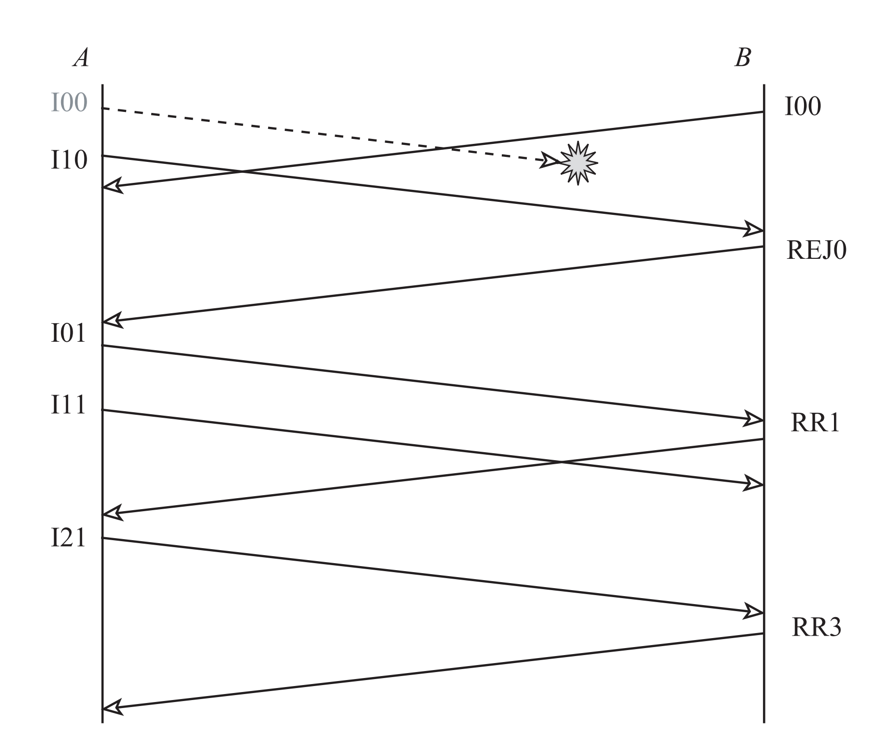
f En reprenant les mêmes hypothèses de travail qu'à la question c, donnez le diagramme correspondant aux différentes trames échangées entre *A* et *B*.

Solution

a Les trames échangées entre les stations sont représentées différemment. Par convention, l'axe des temps est vertical. Une flèche orientée représente une trame, dont la nature est donnée à l'origine de la flèche (côté émetteur). Elle est prise en compte par le récepteur à l'instant figuré par la fin de la flèche (le récepteur réagit alors à la trame qu'il vient de recevoir). À la figure ci-après, les trames erronées sont représentées par des flèches et des informations de commande grisées. La figure 2.16 donne les échanges de trames entre *A* et *B*.

Figure 2.16

Diagramme des échanges avec rejet simple des trames erronées de la question a.

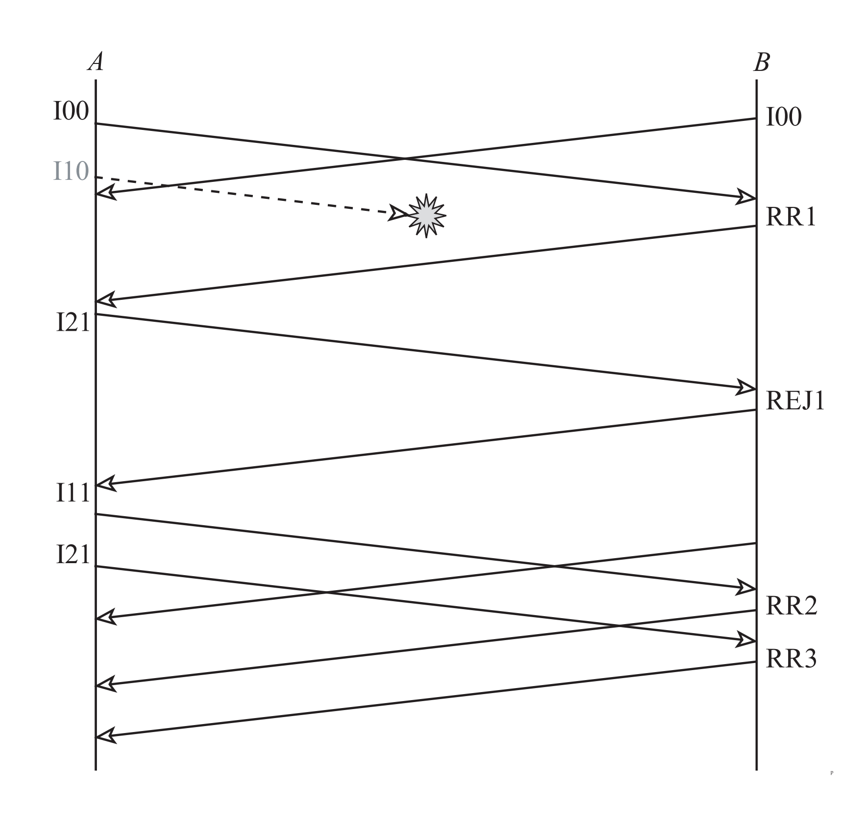


b La figure 2.17 donne les échanges entre *A* et *B* pour la question b.

Figure 2.17

Diagramme des échanges en réponse à la question b.

La fenêtre d'émission de A est fermée après l'émission de I10 et B ne peut pas émettre de trame REJ. A peut émettre I21 dès qu'il a reçu l'acquittement de sa première trame (par RR1)





c La figure 2.18 donne les échanges entre *A* et *B* pour la question c.

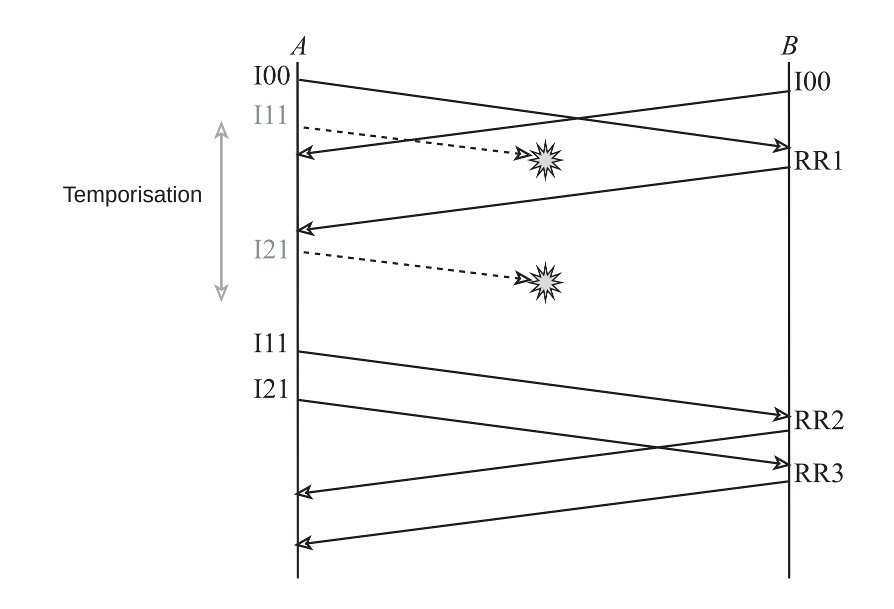


Figure 2.18

Diagramme de réponse à la question c.

A émet la trame I21 dès que sa fenêtre est ouverte par la trame RR1. Comme c'est la dernière trame I à émettre, B ne peut pas la rejeter. Il faut donc que A attende l'expiration du temporisateur associé à I11 pour qu'il puisse réémettre les deux dernières trames.

d La figure 2.19 donne les échanges entre *A* et *B* pour la question d.

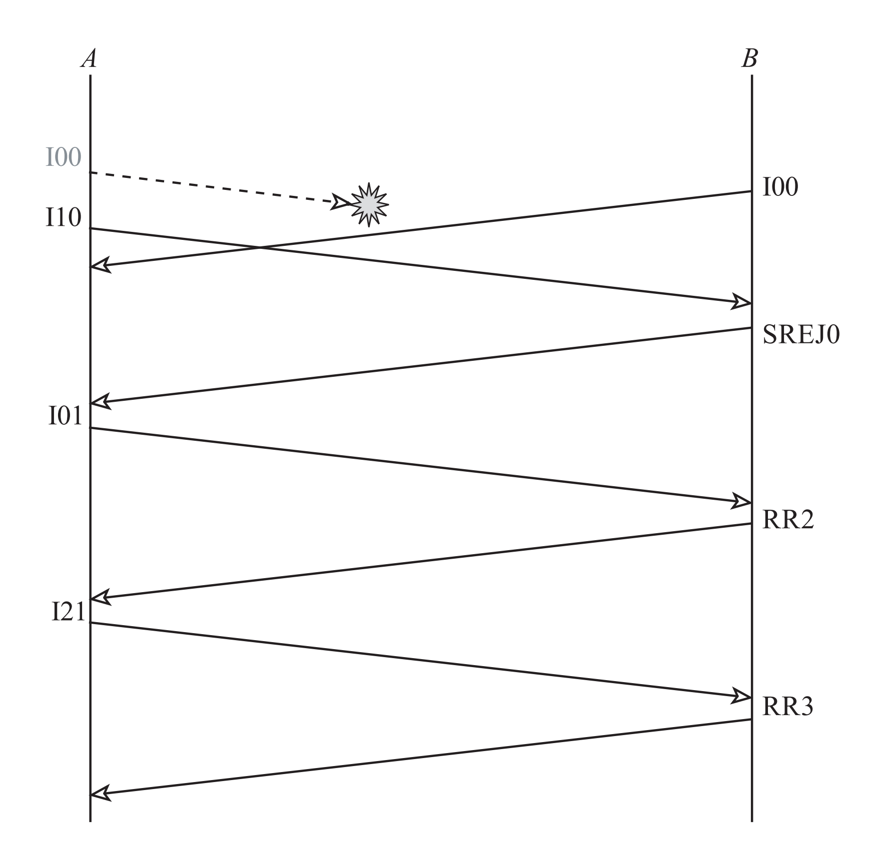


Figure 2.19

Diagramme de réponse à la question d.

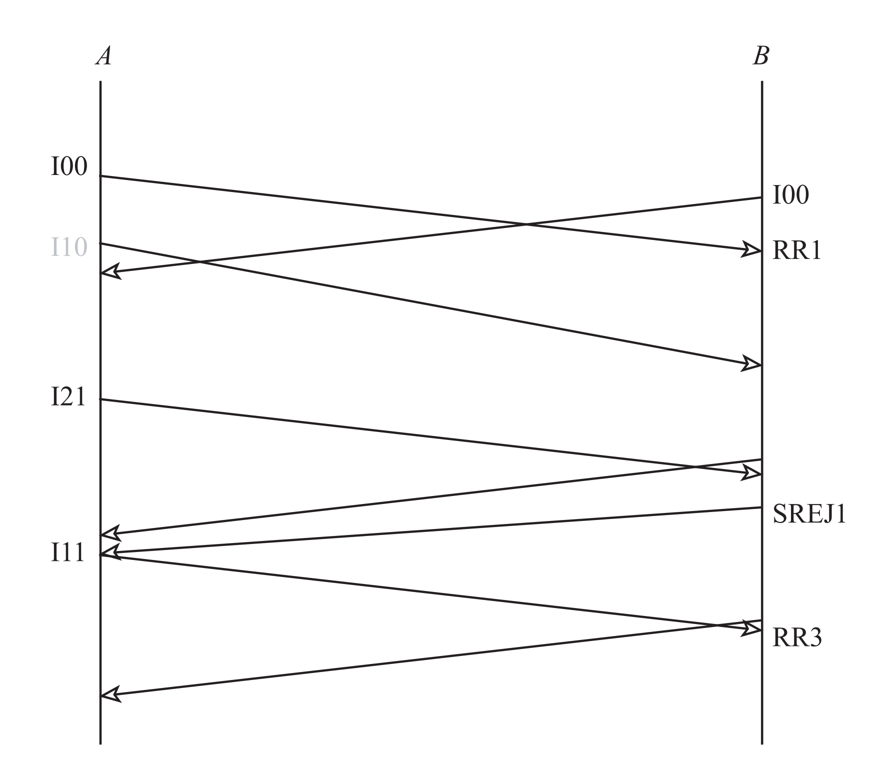
B a détecté la rupture de séquence et redemande la trame I00. Elle sera réémise sous la forme de I01 pour acquitter la trame reçue de B. Lorsque I01 parvient correctement à destination, B peut reconstituer la séquence complète des trames émises et acquitter les deux premières. A émet alors sa dernière trame.

e La figure 2.20 donne les échanges entre *A* et *B* pour la question e.

Figure 2.20

Diagramme de réponse à la question e.

Lorsque B a reçu la trame I11, il peut acquitter toutes les trames qu'il a reçues.

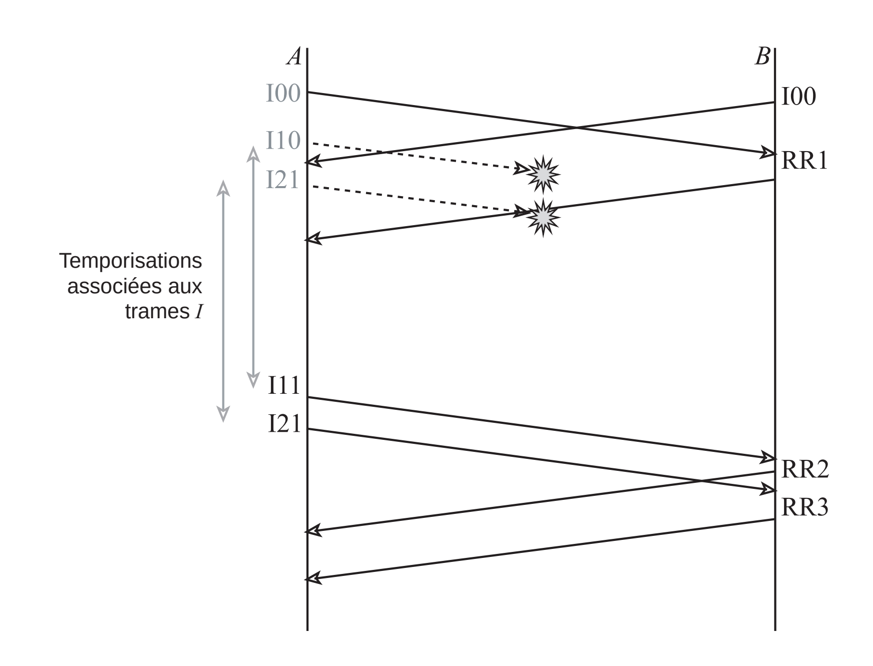


f La figure 2.21 donne les échanges entre *A* et *B* pour la question f.

Figure 2.21

Diagramme de réponse à la question f.

Les deux dernières trames de A étant erronées, toutes les trames de sa fenêtre sont ignorées. Il faut attendre l'expiration des temporisations associées aux trames pour les réémettre.



La réponse à cette question est donnée par le diagramme de la figure 2.21. Nous y remarquons que, comme dans le cas du rejet simple, il faut attendre la reprise sur temporisation pour réémettre les deux dernières trames I.

Remarque

Nous constatons des similitudes et certaines différences entre les deux modes de rejet. Commençons par les similitudes :

1. Dans le rejet sélectif comme dans le rejet simple, seules les trames I sont acquittées explicitement, soit par une trame RR ou RNR, soit par l'incrémentation du compteur *N(R)* inséré dans la trame I qui suit la bonne réception des trames. Cet acquittement valide toutes les trames I jusqu'à la trame de numéro *N(R) – 1* incluse.
2. Une trame I erronée ne se détecte qu'*a posteriori* : par exemple, c'est seulement parce que le récepteur a reçu la trame I10 qu'elle peut détecter que la trame I00 était en erreur. Ainsi, une trame SREJ, comme une trame REJ, ne sera émise qu'après la première trame I reçue correctement mais qui n'est pas en séquence.
3. Le compteur de trames reçues correctes est incrémenté *après* que le récepteur a vérifié que la trame respecte deux conditions : elle est bonne (format et FCS corrects) *et* elle est en séquence (la valeur *N(S)* de la trame reçue est égale à la valeur *N(S)* de la trame précédente augmentée d'une unité).
4. Dans les deux techniques de rejet, la valeur courante de *N(R)* signifie que toutes les trames I jusqu'à la valeur *N(R) – 1 sont bonnes et en séquence*. En ce sens, on peut dire que *N(R)* compte, modulo 8, le nombre de trames *I* bien reçues.

Examinons maintenant les différences :

1. La manière de renvoyer les trames erronées est très différente dans les deux modes : dans le rejet sélectif, on ne renvoie une trame I que si le récepteur de la trame le demande explicitement par une trame SREJ (ou par expiration de la temporisation associée si cette trame est la dernière de la série). La séquence des trames émises va ainsi dépendre du succès de la réémission des trames précédentes. L'incrémentation du compteur *N(R)* dépend de deux conditions : a) la trame I reçue est correcte, b) on peut reconstituer la séquence des trames I en chaînant toutes celles déjà reçues correctement.

   Par exemple, à la figure 2.19, après réémission de la trame I00 (sous la forme I01), l'émetteur continue l'émission avec la trame I21 tant que la fenêtre d'anticipation n'est pas pleine, alors qu'avec le rejet simple il doit réémettre toutes les trames I à partir de la trame erronée.
2. Le rejet sélectif d'une trame signifie seulement que le récepteur a trouvé au moins une trame I qui n'est pas en séquence, ce qui lui permet d'envoyer une trame SREJ. Par exemple, si plusieurs trames I successives sont erronées, le récepteur doit envoyer *une trame SREJ pour chaque trame I erronée*. Ainsi, tant que la trame I00 n'est pas correctement reçue, le compteur *N(R)* ne peut s'incrémenter. Dès que la trame I01 est reçue, le récepteur peut acquitter les trames I01, I11 et I21 comme on peut le constater avec l'envoi de la trame RR3.

EXERCICE 13 AUTRE EXEMPLE DE REJET DES TRAMES ERRONÉES

Énoncé

Soit deux stations émettant en bidirectionnel simultané 4 trames I consécutives et utilisant une procédure de type HDLC. On suppose que l'initialisation de la liaison de données a été effectuée et que le temps de propagation et d'acquittement des trames est négligeable. On suppose de même que la taille de la fenêtre est suffisante pour ne pas bloquer les processus d'émission. Les trames numérotées 0 et 2 sont erronées lorsqu'elles sont émises les deux premières fois dans les deux sens. Elles sont réémises correctement la fois suivante.

a Donnez le diagramme des trames échangées entre les deux stations avec le mode de rejet simple (REJ).

b Donnez le diagramme des trames échangées entre les deux stations lorsqu'elles utilisent le mode de rejet sélectif (SREJ au lieu de REJ).

Solution **a** Le diagramme de la figure 2.22 montre les échanges de trames selon le protocole LAP-B.

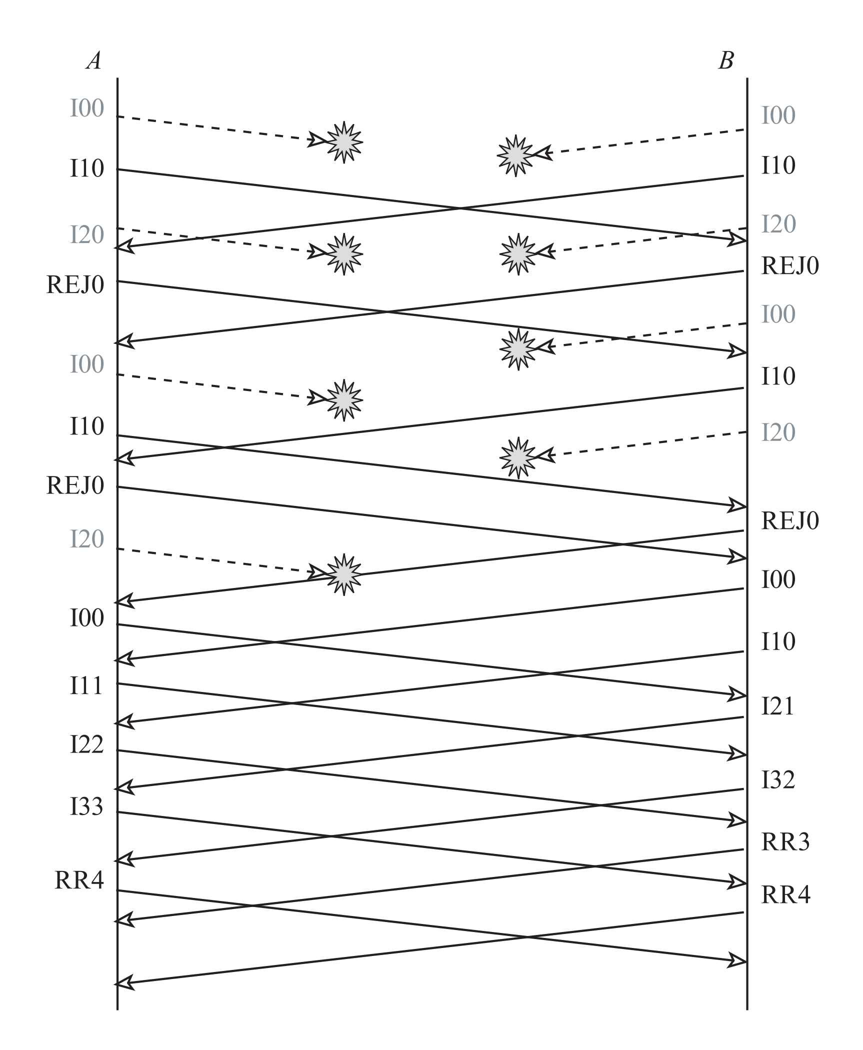


Figure 2.22

Échange de trames selon le protocole LAP-B.

b Le diagramme de la figure 2.23 montre les échanges de trames avec rejet sélectif des trames erronées. L'échange peut se décomposer en quatre phases :

- (1) Envoi des premières trames I par les deux équipements.
- (2) On détecte la rupture de séquence ; *A* et *B* demandent la réémission des trames 0, qui sont à nouveau mal transmises.
- (3) 0 et 1 sont maintenant correctes ; *A* et *B* redemandent les trames n° 2, encore erronées.
- (4) Toutes les trames sont maintenant correctes. Elles sont acquittées par RR4.



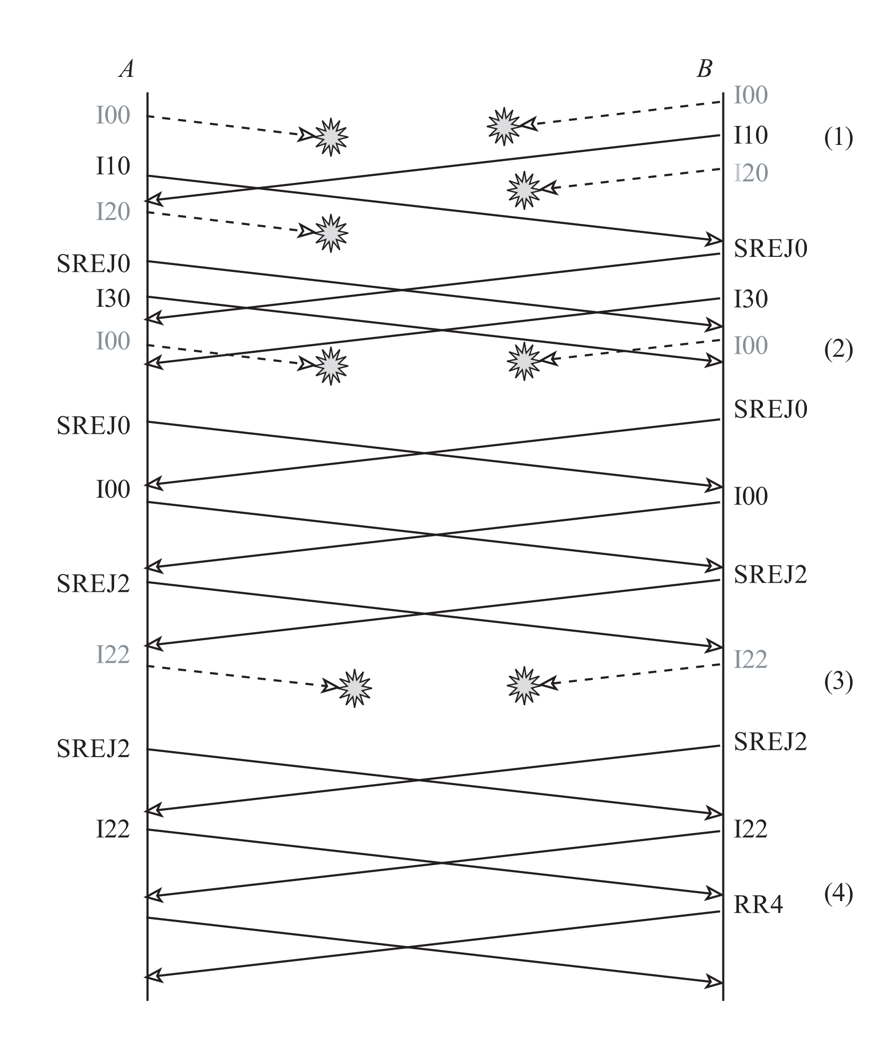


Figure 2.23

Diagramme d'échanges de trames avec rejet sélectif.

Exercice 14 Cas d'équipements ayant des débits binaires différents

Énoncé

Une banque gère les liaisons de données de ses automates bancaires grâce au protocole HDLC avec rejet simple[8]. Chaque automate possède une liaison de données ayant les caractéristiques suivantes :

- débit binaire : 14 400 bit/s ;
- fenêtre d'anticipation égale à 4.

Le serveur de la banque utilise une liaison de données configurée comme suit :

- débit binaire : 2,048 Mbit/s ;
- fenêtre d'anticipation égale à 7.

a Les automates bancaires peuvent-ils savoir que les paramètres de connexion sont différents aux deux extrémités de la liaison ? Où se fait l'adaptation entre les services utilisés aux deux extrémités ?

8. La banque utilise en fait un GFA (groupe fermé d'abonnés), qui permet à un groupe d'abonnés d'utiliser l'infrastructure d'un réseau public de données comme s'ils se trouvaient au sein d'un réseau privé.

Énoncé (suite)

L'introduction de la carte bancaire d'un client lance une opération bancaire, qui se déroule comme suit : après la saisie des données du client, l'automate envoie une trame I décrivant l'opération demandée au serveur de la banque. Celui-ci, après vérification des données envoyées par l'automate, lance l'application de gestion du compte client, puis l'application fabrique un accusé de réception et l'envoie à l'automate dans une trame I. L'automate pourra alors imprimer le ticket à fournir au client (ou afficher le résultat de l'opération sur l'écran de l'automate).

b Décrivez l'échange de trames entre l'automate bancaire et son point d'accès au réseau, en supposant que la transmission des données s'effectue avec une erreur de transmission dans la première trame I émise dans chaque sens (on ne s'intéresse pas à la gestion du bit *P/F*).

c Cette fois, le traitement des erreurs de transmission se fait par rejet sélectif (SREJ) des trames (nous n'utilisons plus le protocole LAP-B). En prenant les mêmes hypothèses que précédemment, donnez le diagramme décrivant les échanges entre l'automate et son point d'accès au réseau.

Solution

a Non, les automates bancaires – pas plus que le serveur – ne peuvent le savoir, car ils ont choisi leurs paramètres de connexion en fonction de leurs propres besoins et possibilités. L'adaptation entre les paramètres de l'équipement local et ceux de l'équipement distant se fait donc dans le nœud d'accès au réseau sur lequel chaque équipement est connecté.

b Les trames échangées sont données à la figure 2.24.

Une seule trame I suffit pour contenir les informations du client. Puisqu'elle est erronée, le récepteur ne peut ni l'acquitter, ni la rejeter puisqu'elle n'est pas suivie d'une trame I hors séquence. C'est grâce à la temporisation associée à la trame que l'émetteur constate qu'elle n'a pas été acquittée. Il va donc la réémettre[9].

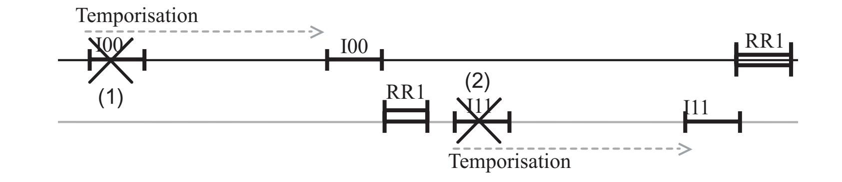


Figure 2.24

Diagramme des trames échangées entre l'automate et son point d'accès.

1 : trame d'informations contenant les informations du client.

2 : trame d'informations contenant les résultats de transaction à afficher ou à imprimer.

c Les hypothèses de travail étant les mêmes qu'à la question précédente, la reprise sur erreur se fera de la même façon, puisqu'on ne peut pas utiliser de trame de rejet… Les échanges de données seront identiques à ceux de la figure 2.24.

9. En fait, la trame sera réémise avec le bit *P = 1* et l'acquittement portera le bit *F = 1*.



Les concepts généraux des réseaux

L'opérateur d'un réseau comptant un grand nombre d'abonnés doit offrir une garantie de bon fonctionnement, tout en optimisant les coûts d'évolution et de maintenance du réseau. Il lui faut donc exploiter une infrastructure de réseau conçue pour optimiser les ressources mises en œuvre. Pour cela, on utilise principalement deux techniques de commutation : la commutation de circuits et la commutation de paquets. La première, utilisée pour les communications téléphoniques, réserve des ressources physiques pour chaque couple d'équipements désirant communiquer. La seconde est la plus utilisée pour les échanges de données informatiques. Pour assurer le transfert des données, deux services réseau sont possibles : l'un, évolué, en mode connexion et l'autre, plus simple, sans connexion. Le premier, normalisé sur le plan international, est baptisé *circuit virtuel*. Il transporte des unités de données appelées *paquets*. Le second, transportant des datagrammes, sert à l'échelle mondiale dans Internet. Des fonctions de contrôle interne du réseau assurent la meilleure gestion des ressources disponibles, pour en garantir le meilleur usage possible aux utilisateurs.

1 Infrastructure des réseaux de communication

Jusqu'à présent, nous avons examiné la manière dont les deux extrémités d'une liaison de données s'échangent les données. Pour connecter un grand nombre d'utilisateurs, il devient très vite irréaliste de les interconnecter deux par deux. Il faut donc généraliser l'échange entre deux équipements à un ensemble de *N* équipements. Le terme générique d'*équipements terminaux* désigne aussi bien des ordinateurs que de simples téléphones. Nous étudions dans la suite les matériels et les procédures à mettre en place pour permettre un dialogue entre deux équipements quelconques de cet ensemble.

En effet, comme le montre la figure 3.1, pour permettre toutes les communications au sein d'un ensemble de *N* équipements terminaux, il faudrait $N(N-1)/2$ liaisons ; chaque équipement devrait alors gérer $N-1$ liaisons. Concrètement, pour faire dialoguer directement 100 stations, il faudrait environ 5 000 liaisons ! Cela montre l'impossibilité d'envisager des liaisons distinctes et exclusives entre deux équipements terminaux connectés à un grand réseau. Il est donc nécessaire de fédérer les moyens de communication et de les partager entre tous les équipements terminaux pour constituer un *réseau de communication*. Ce terme possède différentes significations suivant le contexte : il peut comprendre l'ensemble des ressources (les équipements terminaux inclus) ou ne désigner que le réseau de communication, parfois encore appelé *réseau de transport*. Dans ce chapitre, le réseau de communication n'englobe pas les équipements terminaux.

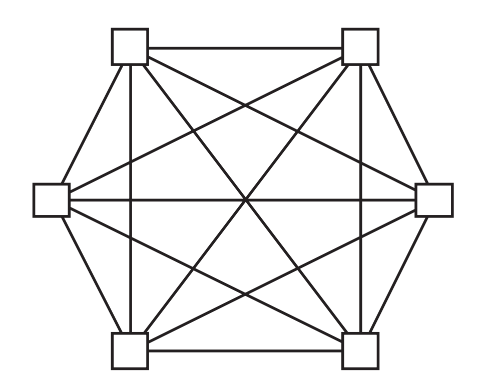
Parmi les équipements terminaux, on distingue habituellement deux catégories : les ETTD (équipement terminal de traitement de données) et les ETCD (équipement de terminaison du circuit de données). Les ETTD sont les équipements d'extrémité abonnés au réseau de communication (un ordinateur, une imprimante, un fax…) alors que les ETCD servent de point d'entrée dans le réseau aux ETTD. Nous allons utiliser également le terme de « nœud d'accès » pour désigner les ETCD.

Définition

Un réseau de communication est constitué d'un ensemble de liaisons de données et de nœuds. Il constitue l'ensemble des ressources mises à la disposition des équipements terminaux pour échanger des informations.

Figure 3.1

Connexions directes entre N utilisateurs : il y a N(N – 1)/2 liaisons.





1.1 COMMUNICATIONS DANS LES RÉSEAUX

La présence d'une multitude d'équipements terminaux oblige à définir un système d'identification cohérent au sein du réseau pour les différencier : c'est la fonction d'*adressage.* La fonction de *routage* permet d'acheminer une information vers un destinataire dans tout le réseau de communication, selon son adresse (voir figure 3.2).

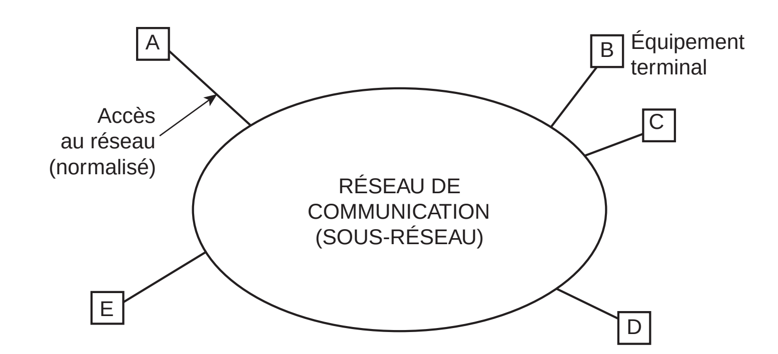


Figure 3.2

Structure d'un réseau de communication.

Quand un *opérateur* possède un réseau de communication, il met cette ressource à la disposition de tiers, moyennant rétribution (c'est le cas des grands réseaux : France Télécom gérait entièrement le réseau téléphonique français jusqu'en 1998). L'organisation du réseau est alors du ressort exclusif de l'opérateur. Son accès est normalisé, tant pour les caractéristiques mécaniques et électriques que pour les procédures de dialogue. Il est indépendant de la structure interne du réseau : par exemple, dans le cas des réseaux radiomobiles, il se fait par transmission sur la voie hertzienne alors que les transmissions au sein du réseau se font sur des liaisons filaires.

Les *réseaux locaux d'entreprise*, appelés LAN (*Local Area Network*), sont généralement la propriété exclusive de l'utilisateur. Celui-ci gère et maintient alors entièrement tous les équipements et les moyens de communication. Les LAN accueillent plusieurs centaines d'équipements sur une distance de quelques kilomètres. La ressource partagée entre les équipements est le support de transmission qui assure la *diffusion* (*broadcast*) : tous les équipements sont reliés au support commun et tout message émis est reçu par l'ensemble des équipements. Cette caractéristique nécessite des architectures spécifiques qui seront traitées au chapitre 5.

1.2 RÉSEAUX À COMMUTATION

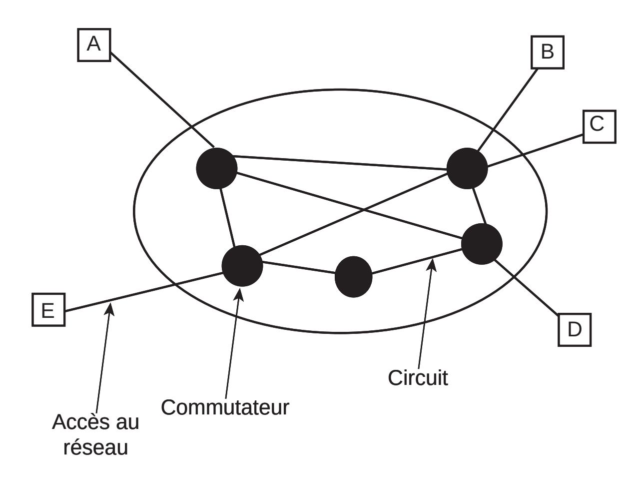
Les réseaux grande distance, appelés aussi WAN (*Wide Area Network*), relient plusieurs centaines de milliers, voire des millions d'équipements terminaux sur un territoire national ou à l'international. Il n'est donc pas possible de partager le même support de transmission, ni de raccorder directement deux abonnés désirant communiquer. On crée une structure de communication qui, en mettant bout à bout des tronçons de lignes raccordés par un ensemble de *commutateurs*, réalise une connexion entre deux abonnés d'un réseau ; on parle alors de *réseau à commutation*. De ce fait, un réseau à commutation fournit l'équivalent d'une liaison de données point à point entre deux équipements terminaux quelconques abonnés au réseau.

Des commutateurs, qui ont pour fonction de concentrer, d'éclater et de rediriger les informations, relient les équipements terminaux. Ils communiquent entre eux par des *circuits* point à point, qui constituent les artères de communication du réseau. On considère un

réseau de communication comme un graphe, où les nœuds représentent les commutateurs et les arcs figurent les circuits (quelquefois appelés canaux, jonctions, lignes de transmission ou même liaisons, selon les cas). La figure 3.3 montre la structure d'un réseau à commutation.

Figure 3.3

Structure générale d'un réseau à commutation : commutateurs et circuits.



La *topologie* du réseau définit la façon de relier les différents commutateurs. Dans une topologie *en étoile* (voir figure 3.4a), un même commutateur central relie l'ensemble des commutateurs. Certaines fonctions, comme le routage, sont alors très simples. Un tel réseau est cependant très fragile car il dépend essentiellement du bon fonctionnement du commutateur central.

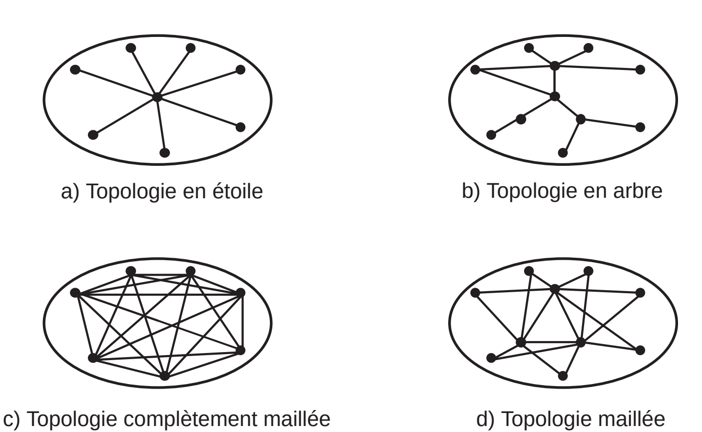
La généralisation du cas précédent, avec introduction d'une hiérarchie, donne la topologie *en arbre* de la figure 3.4b : chaque commutateur est relié à un ensemble de commutateurs du niveau inférieur. Dans les topologies en arbre ou en étoile, il n'y a toujours qu'un chemin possible entre deux commutateurs : toute rupture de liaison entre eux empêche le dialogue entre certains équipements terminaux.

Dans la topologie *complètement maillée* de la figure 3.4c, chaque commutateur est relié à tous les autres. On atteint alors un haut niveau de sécurité, au prix d'une augmentation considérable du nombre de liaisons et, par conséquent, du coût du réseau.

Dans la plupart des grands réseaux, la solution choisie est un mélange des solutions précédentes : le réseau est hiérarchisé selon une topologie en arbre et utilise un certain degré de maillage. La figure 3.4d montre un exemple de réseau *maillé*.

Figure 3.4

Différentes topologies d'un réseau à commutation : un maillage plus ou moins dense.



Remarque

La fonction de routage prend une importance particulière dans un réseau à commutation puisqu'en règle générale il n'y a pas de lien direct entre équipements terminaux, mais une multitude de chemins possibles qui traversent plusieurs commutateurs et empruntent plusieurs liaisons.

Commutation de circuits

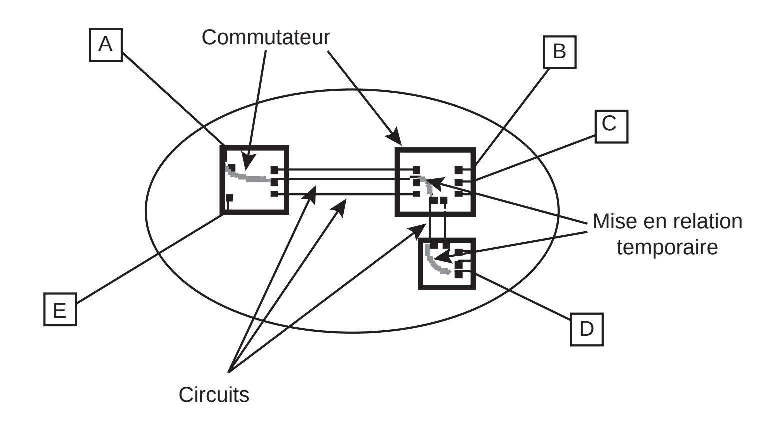
Dans les réseaux à commutation de circuits, de multiples supports de transmission relient les différents commutateurs. Échanger des informations entre deux équipements terminaux nécessite de déterminer un chemin dans le réseau et de réserver un support de transmission entre chaque paire de commutateurs situés sur ce chemin. Chaque commutateur reçoit les signaux d'une liaison et les retransmet sur la liaison vers le commutateur suivant. Le réseau téléphonique est l'exemple le plus connu de réseau à commutation de circuits. En téléphonie, le mot *circuit* désigne une liaison entre deux commutateurs.

Tout dialogue entre équipements terminaux se décompose en trois phases. La première, l'*établissement du circuit*, réserve l'ensemble des circuits nécessaires à l'intérieur du réseau. Suit la phase classique de *transfert des informations*. Enfin, la phase de *libération* rend les différents circuits utilisés disponibles pour les communications ultérieures. La libération se fait à la demande de l'un des équipements terminaux (ou par le réseau s'il détecte qu'un des équipements du chemin est en panne). Tant que la libération n'a pas eu lieu, les circuits restent attribués aux mêmes correspondants, même s'ils n'effectuent aucun transfert d'informations sur une longue durée.

Ce type de commutation présente l'inconvénient de monopoliser les circuits entre commutateurs pendant toute la durée du dialogue, même pendant les périodes de silence. Il est donc nécessaire de multiplier les circuits entre commutateurs ; on parle dans ce cas de *faisceaux* (*trunks*). De plus, la commutation de circuits requiert la disponibilité simultanée des deux équipements terminaux pour tout dialogue. En revanche, elle présente l'avantage d'être assez simple et peut s'employer sur un réseau analogique ou numérique. Dans le cas d'un réseau numérique, la mémoire nécessaire dans les commutateurs est réduite. La figure 3.5 décrit le principe de fonctionnement d'un réseau à commutation de circuits. Nous voyons que la communication entre *A* et *D* traverse différents commutateurs et emprunte plusieurs circuits. Les deux équipements terminaux disposent de l'ensemble de ces ressources pour la durée de leur communication.

Figure 3.5

Principe de la commutation de circuits : réservation de ressources physiques pour la durée de la communication.



Commutation de messages

La commutation de messages est la première technique imaginée pour les réseaux transportant des données informatiques. Un *message* se définit comme une suite de données binaires formant un tout cohérent pour les utilisateurs (une page de texte, un fichier son, une image fixe ou animée…).

Un utilisateur qui veut émettre un message l'envoie au commutateur en précisant l'adresse du destinataire. Le commutateur attend la réception complète du message, le stocke, analyse l'adresse du destinataire puis émet le message vers le commutateur voisin adéquat ou, le cas échéant, vers l'équipement terminal (technique *store and forward*). L'aiguillage du message s'effectue en fonction des informations de contrôle. Le commutateur conserve le message si la liaison est occupée : chaque commutateur se comporte donc comme une mémoire tampon.

Le message transite ainsi à travers le réseau par émissions successives entre les commutateurs jusqu'au destinataire. La figure 3.6 montre l'infrastructure d'un réseau à commutation de messages : *A*, *B*, *C*, *D* et *E* sont des abonnés au réseau qui échangent des messages et les carrés représentent les commutateurs. Dans la commutation de messages, les liaisons ne sont utilisées que pour la durée de transmission entre les deux équipements adjacents. Chaque commutateur doit être capable de stocker le message entier.

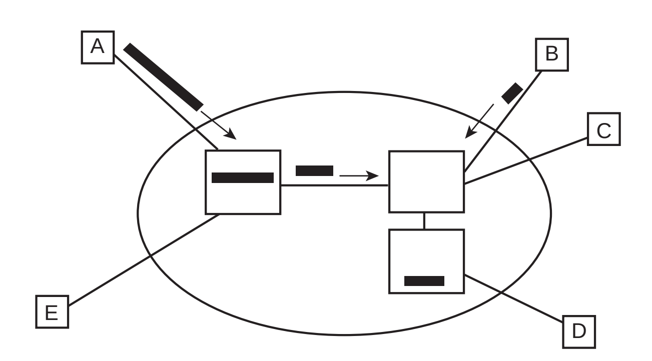


Figure 3.6

Réseau à commutation de messages : les liaisons ne sont utilisées que pour la durée de transmission entre équipements adjacents.

Comme un commutateur gère simultanément plusieurs échanges, la taille de la mémoire nécessaire à la gestion des messages est importante et entraîne des problèmes d'allocation complexes. De plus, les liaisons entre commutateurs ne sont pas d'une fiabilité totale. Des protocoles de liaison de données sont nécessaires entre chaque paire d'équipements. Les commutateurs gèrent autant de liaisons de données que d'équipements auxquels ils sont reliés.

Le délai de transmission dans le réseau est fonction du nombre de commutateurs traversés et de la longueur du message. Remarquons que la probabilité d'une erreur sur un message augmente avec sa longueur : la transmission de messages longs dans le réseau est très pénalisante. Une amélioration de cette technique réduit la taille des messages envoyés et conduit à la *commutation de paquets*.

Commutation de paquets

Dans la commutation de paquets, on découpe d'abord le message en plusieurs morceaux, appelés *paquets*, avant de l'envoyer dans le réseau : cela s'appelle la *fragmentation*. Comme dans un réseau à commutation de messages, les commutateurs utilisent des informations de contrôle pour acheminer correctement les paquets depuis l'expéditeur jusqu'au destinataire.



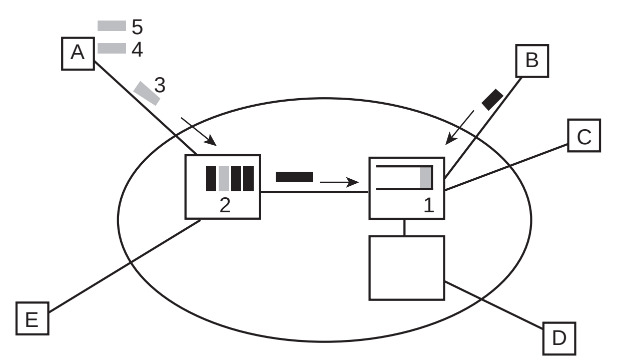
L'opérateur du réseau (ou des normes internationales) définit le format de l'en-tête et la taille maximale d'un paquet. Le destinataire doit attendre la réception de tous les paquets pour reconstituer le message et le traiter : cette opération est le *réassemblage*.

Un paquet ne forme donc pas un tout logique pour l'équipement terminal : ce n'est qu'un « élément d'information », acheminé dans le réseau par les réémissions successives entre commutateurs. Sa petite taille réduit le délai global d'acheminement des messages. Cependant, elle accroît la complexité de sa gestion dans les commutateurs : le dimensionnement de la mémoire des commutateurs est un élément important dans la détermination de la capacité et les performances d'un réseau à commutation de paquets. Si la mémoire d'un commutateur est entièrement utilisée, celui-ci n'est plus en mesure de recevoir de nouveaux paquets. Il peut, dans certains cas, détruire des paquets et dégrader les performances du réseau. L'ensemble des techniques mises en œuvre pour éviter la saturation de la mémoire des commutateurs s'appelle le *contrôle de congestion*. (La description des techniques de contrôle de congestion dépasse le cadre de cet ouvrage.)

Comme pour la commutation de messages, une paire d'équipements ne monopolise plus une liaison entre commutateurs : celle-ci supporte la transmission de paquets de multiples utilisateurs. Si le débit de la liaison est supérieur au flux transmis par l'ensemble des utilisateurs, elle peut supporter de plusieurs dialogues simultanés tout en donnant à chaque utilisateur l'impression d'être seul sur le réseau. Ainsi, même si le flux généré par un utilisateur donné augmente subitement, l'impact sera faible sur le flux global dans le réseau. La figure 3.7 montre comment un message constitué de cinq paquets est transmis d'un utilisateur à l'autre.

Figure 3.7

Réseau à commutation de paquets : il peut y avoir simultanément transmission de plusieurs paquets d'un même message sur différentes liaisons du réseau.



Le message émis par A est découpé en 5 paquets, acheminés un par un par le réseau.

Remarque

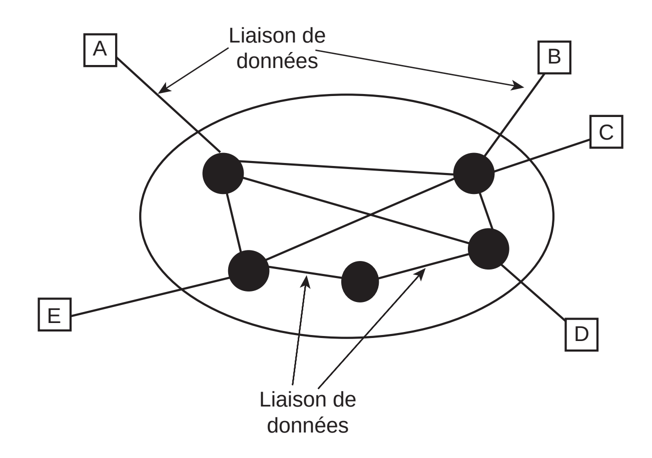
La méthode pour structurer les équipements et les services est hiérarchique : à chaque extrémité d'un circuit on trouve une succession d'entités assurant un service donné pour une entité de niveau supérieur.

À une extrémité de la liaison de données, l'*entité de liaison* assure un dialogue fiable avec l'entité de même type située à l'autre extrémité. Une entité de niveau supérieur, l'*entité de réseau*, assure le routage des paquets à travers le réseau. Elle utilise pour cela l'entité de liaison comme une « boîte noire » lui fournissant un service. Elle est la seule à interpréter et exploiter l'en-tête pour acheminer les paquets jusqu'à leur destination. L'entité de liaison de données considère le paquet comme l'élément à transmettre : par exemple, elle l'insère dans le champ d'informations de la trame gérée selon le protocole HDLC. Ce procédé s'appelle l'*encapsulation*.

La figure 3.8 montre la mise en œuvre d'une succession de liaisons de données, et la figure 3.9, le découpage d'un message en paquets et leur intégration dans plusieurs trames.

Figure 3.8

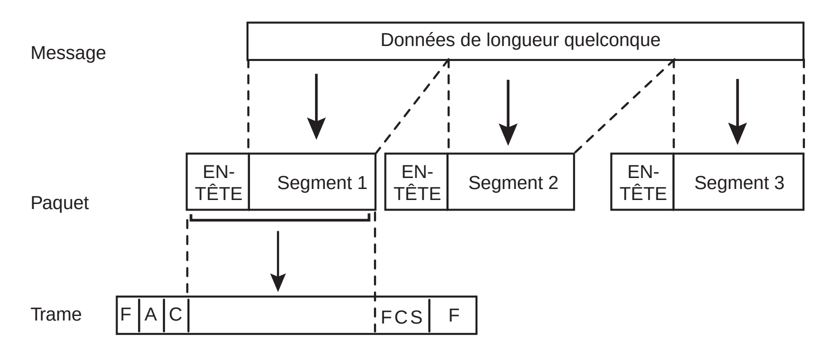
Mise en œuvre des liaisons de données dans un réseau à commutation : des successions de liaisons de données relient toutes les paires d'équipements.



Une succession de liaisons de données est mise en œuvre entre toutes les paires d'équipements pour acheminer les données de l'expéditeur jusqu'au destinataire.

Figure 3.9

Découpage d'un message en trois paquets insérés dans trois trames successives.



Le message est coupé en trois morceaux : un en-tête s'ajoute à chacun pour constituer un paquet, transmis dans le champ Information d'une trame.

Commutation de cellules

Une nouvelle technique de commutation émerge dans les années 1990, tout particulièrement pour le RNIS (réseau numérique à intégration de services) large bande. Il s'agit d'une commutation hybride, utilisant une technique dite ATM (*Asynchronous Transfer Mode*). Les informations sont toutes découpées en petits paquets de taille fixe, les *cellules*, qui optimisent la gestion interne du réseau et de ses commutateurs. Chaque cellule contient 53 octets : un en-tête de 5 octets suivi de 48 octets d'informations utiles.

L'objectif est de transmettre en temps réel sur le même réseau des données, de la parole et des images. Les problèmes posés par les méthodes de commutation précédentes sont multiples : trop faible capacité des réseaux, faibles vitesses de transmission, rigidité des services offerts, interconnexion difficile entre différents types de réseaux, coûts élevés des solutions performantes, incapacité de transmettre paroles et images en temps réel, qualité de service insuffisante. ATM est une solution à l'intégration de services, elle utilise à la fois la commutation de circuits avec une notion de réservation de ressources et celle de commutation de paquets avec le découpage des informations en cellules.



La technique ATM exploite le fait que les supports sont des fibres optiques, donc des supports de très bonne qualité : elle n'assure pas la détection des erreurs sur les données et réduit les contrôles à la seule vérification de l'en-tête des cellules. Pour une communication donnée, les cellules passent toutes par le même *chemin virtuel* et leur traitement, effectué dans les commutateurs du réseau, est réduit au strict minimum. ATM offre ainsi une excellente qualité de service et apporte un confort d'utilisation tel, que la qualité des liaisons établies à la demande est la même que celle des liaisons permanentes. Par ailleurs, les délais de traversée du réseau sont garantis très faibles (commutateurs puissants et dimensionnés pour le traitement de cellules de taille fixe).

ATM permet la mise en œuvre d'applications d'images avec compression en temps réel et constitue l'unique technologie pour le multimédia haut débit. On la rencontre aujourd'hui dans le cœur des réseaux d'opérateurs de télécommunications. Son coût très élevé a freiné son déploiement et les technologies Gigabit Ethernet (voir chapitre 5) sont une alternative intéressante pour les réseaux locaux.

1.3 Optimisation des ressources de transmission : le multiplexage

La gestion des ressources d'un grand réseau est une tâche lourde qui nécessite des investissements considérables de la part de l'opérateur. Ce dernier a donc le souci d'optimiser son infrastructure, afin d'offrir à moindre coût un service satisfaisant à ses clients. En plus des techniques de commutation vues aux sections précédentes, il utilise des techniques de multiplexage pour minimiser le nombre d'artères nécessaires.

Lorsque la bande passante d'un support est nettement plus large que le spectre du signal à transmettre, il est intéressant d'utiliser ce support pour transmettre simultanément plusieurs communications ; on parle alors de *multiplexage*. Le *démultiplexage* consiste à reconstituer et à redistribuer les différents signaux sur les bonnes artères à partir du signal multiplexé. Deux techniques principales fonctionnent : le *multiplexage fréquentiel* ou *spatial* et le *multiplexage temporel*.

Multiplexage fréquentiel ou spatial

Le multiplexage fréquentiel ou spatial s'utilise dans les transmissions analogiques comme dans les transmissions numériques. Il consiste à transposer en fréquence – en utilisant une technique de modulation – les *n* signaux d'entrée, chacun avec une fréquence porteuse différente, en les juxtaposant dans la bande des fréquences utilisables. On parle alors d'AMRF (accès multiple à répartition en fréquence) ou de FDMA (*Frequency Division Multiple Access*). La figure 3.10 montre comment multiplexer *n* voies utilisant la même bande passante sur un support à large bande.

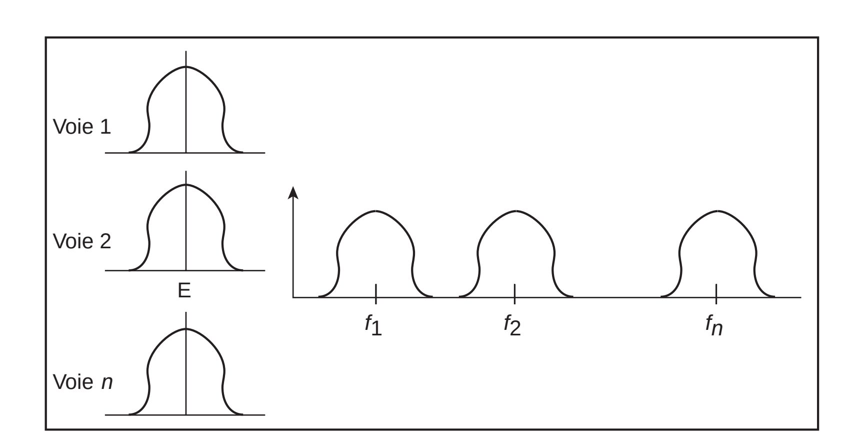


Figure 3.10

Multiplexage en fréquence : juxtaposition de signaux dans la bande des fréquences utilisables.

Le multiplexage fréquentiel s'utilise aussi sur fibre optique ; on parle alors de multiplexage *en longueur d'ondes*. Les opérations de multiplexage et de démultiplexage se font de manière totalement optique, en jouant sur les phénomènes de réfraction, qui dépendent des longueurs d'onde : un démultiplexeur opère en fait comme un prisme.

Multiplexage temporel

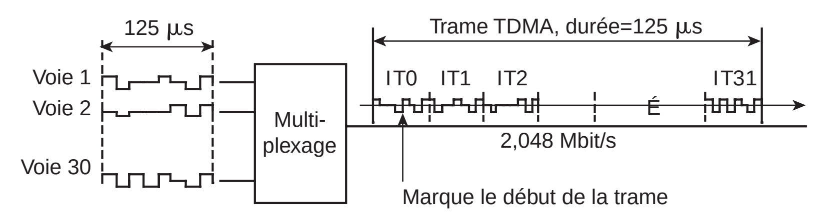
Le multiplexage temporel, appelé AMRT (accès multiple à répartition dans le temps) ou TDMA (*Time Division Multiple Access*), s'utilise dans les transmissions numériques. Si on considère *n* signaux numériques transportés sur *n* voies, le multiplexage temporel revient à transmettre sur un seul support (dit *liaison multiplex*) et dans une même *trame* un élément d'information de chaque voie d'entrée (un bit ou un caractère), pendant un intervalle de temps *IT*. L'intervalle de temps choisi dépend du débit binaire de la liaison multiplex.

Pour constituer une *trame multiplex*, on place dans le premier IT l'élément d'information de la voie 1, l'élément d'information de la voie 2 dans l'IT suivant, et ainsi de suite jusqu'à la voie *n*. Ce cycle est répété indéfiniment dans le temps. Si le débit binaire des voies d'entrée vaut *b* bit/s, alors le débit de la liaison multiplex est de *n**b* bit/s. Pour démultiplexer correctement, il faut, en plus des données des *n* voies, transmettre des éléments de synchronisation afin de réaffecter le bon élément d'information à la bonne voie.

L'AMRT s'utilise dans le réseau téléphonique pour la transmission des communications numérisées. Un signal à 64 kbit/s numérise la parole humaine qui se trouve ainsi transmise à raison d'un octet toutes les 125 µs et codée en bande de base. En Europe, on multiplexe temporellement 30 communications téléphoniques pour constituer un *groupe primaire* ou *canal E1*. La trame multiplex contient 30 IT, chacun contenant l'élément d'information d'une communication (ici un octet), auxquels il faut ajouter l'élément de synchronisation (dans l'IT 0) et un élément de signalisation (en général dans l'IT 16). Pour transporter ces 30 communications, il faut donc un total de 32 IT, soit un débit brut de : 32*64 = 2 048 kbit/s. On parle alors de MIC (modulation par impulsions codées) 30 voies (voir figure 3.11).

Figure 3.11

Multiplexage temporel de 30 voies téléphoniques : juxtaposition dans le temps, octet par octet.



Il est possible de multiplexer des signaux déjà multiplexés si le support de transmission peut transmettre des débits encore plus élevés. Dans le cadre du réseau téléphonique européen, la *hiérarchie de multiplexage* définit différents niveaux de multiplexage, par multiples de 2,048 Mbit/s. Les États-Unis et le Japon emploient une hiérarchie de multiplexage différente.

Remarque

Le multiplexage de *n* signaux, occupant chacun une largeur de bande *B*, génère un signal de largeur supérieure ou égale à *n*B*, pour le multiplexage spatial comme pour le multiplexage temporel. Si on ne réalise aucune économie sur la bande passante consommée, on minimise le matériel nécessaire au transport des données dans le réseau (utilisation d'un seul support de transmission au lieu de *n*). Cette économie est intéressante pour l'opérateur car le coût d'installation et de maintenance d'un support ne dépend pas de son débit mais du lieu de son implantation (en zone urbaine ou rurale) : le prix est le même pour installer des artères à bande passante faible ou large, pour une zone donnée.



2 Notion d’adressage dans les réseaux

Si deux équipements reliés directement l’un à l’autre par une liaison de données n’ont pas de problème d’identification du correspondant, il n’en va pas de même pour les abonnés des grands réseaux : chacun d’eux doit posséder une identité unique, afin que les commutateurs acheminent les données au bon destinataire. Nous avons remarqué précédemment qu’il existait une structure hiérarchique des équipements : la communication utilise un circuit de données, contrôlé par le protocole de liaison, sur lequel sont véhiculés les messages découpés en paquets.

Une entité distincte gère chaque niveau. Il faut, pour chaque niveau, une identification unique de la ressource gérée ; on utilise donc plusieurs niveaux d’adressage. Selon ses besoins, chaque niveau d’adressage doit faire la correspondance entre l’adresse qu’il utilise et les adresses manipulées par les niveaux qui lui sont immédiatement inférieurs ou supérieurs. On distingue principalement trois types d’adresses : *physique*, *logique* et *symbolique*. Présentons-les successivement.

2.1 Adresse physique

L’adresse physique est l’adresse de l’équipement situé au plus près du support de transmission. Elle identifie l’interface série utilisée pour l’émission et la réception des données. Elle distingue, parmi plusieurs interfaces série disponibles, celle vers laquelle émettre ou depuis laquelle sont reçues des données. Elle a une signification purement locale à l’équipement.

En général, les abonnés d’un réseau à commutation n’utilisent guère l’adresse physique, puisqu’une seule liaison point à point les relie au commutateur d’entrée dans le réseau. En revanche, l’adresse physique est indispensable aux commutateurs qui doivent décider sur quelle liaison acheminer les données d’un abonné – ou d’un commutateur – à l’autre.

L’adresse physique est utile dans les réseaux locaux. Par exemple, on identifie avec elle la carte Ethernet qui sert d’accès au support commun du réseau local. Nous y reviendrons au chapitre 5.

2.2 Adresse logique

Pour atteindre un utilisateur quelconque depuis n’importe quel point du réseau, il ne suffit pas de distinguer localement les différentes liaisons disponibles. Il faut que les commutateurs puissent abouter les liaisons à emprunter pour relier la source à la destination. Pour cela, ils doivent identifier un utilisateur parmi tous les usagers du réseau : chaque utilisateur doit donc posséder une adresse unique, connue de tous les commutateurs traversés, à partir de laquelle les points d’accès au réseau organisent le *routage* pour acheminer les données le plus efficacement possible. L’adresse utilisée doit être unique et dépend de la nature du réseau de transport et du mode d’acheminement des données : c’est l’adresse logique. Elle est déterminée par l’opérateur du réseau ou par un organisme international.

L’*adresse IP* utilisée dans Internet en est l’exemple le plus connu. Nous y reviendrons lors du chapitre 6, consacré au réseau Internet et son protocole IP.

2.3 Adresse symbolique

L’adresse logique identifie tous les équipements du réseau. Un utilisateur peu familier des contraintes imposées par la structure du réseau peut avoir des difficultés à mémoriser

cette information. Pour faciliter son accès au réseau, il se choisit (ou l'administrateur du réseau choisit pour lui) une adresse symbolique plus facilement compréhensible et mémorisable qu'une adresse logique[1]. Ainsi par exemple, plutôt que de se souvenir de l'adresse IP : `195.122.1.25`, il retiendra plus facilement l'adresse symbolique : *prenom.nom@mon_fournisseur.mon_pays*… Tout comme l'adresse logique, elle doit être unique pour le réseau. Des organismes internationaux ont proposé une structuration des adresses symboliques, pour garantir leur unicité. Le logiciel gérant la connexion réseau de l'ordinateur au fournisseur d'accès à Internet doit apparier adresse logique et adresse symbolique et mémoriser ces informations.

Remarque

On pourrait faire un parallèle entre adresse symbolique/nom de personne et adresse IP/numéro de téléphone. Une personne, identifiée par son nom, est utilisatrice du réseau téléphonique qui l'identifie par son numéro de téléphone. Un service d'annuaire fait la correspondance entre le nom de la personne et son numéro de téléphone. La comparaison est limitée dans la mesure où les noms de personnes ne sont pas uniques !

3 Notion de service dans un réseau à commutation

On distingue deux types de services réseau : le *service sans connexion* et le *service en mode connecté*, encore appelé *service orienté connexion*. Le premier type est utilisé dans Internet ; le second est proposé dans les réseaux publics de données respectant les normes *X.25* de l'ITU. Ces services correspondent à deux façons d'exploiter la commutation de paquets.

Dans un service sans connexion, l'expéditeur traite chaque paquet comme une unité de données totalement indépendante des autres. Un paquet doit donc inclure l'adresse complète du destinataire, éventuellement celle de l'expéditeur. À tout moment, l'équipement terminal peut fournir au réseau un paquet à transmettre sans procédure préalable. Un tel service est par exemple celui fourni par le réseau postal : une lettre peut être postée à tout moment. La figure 3.12 donne un exemple de service réseau sans connexion.

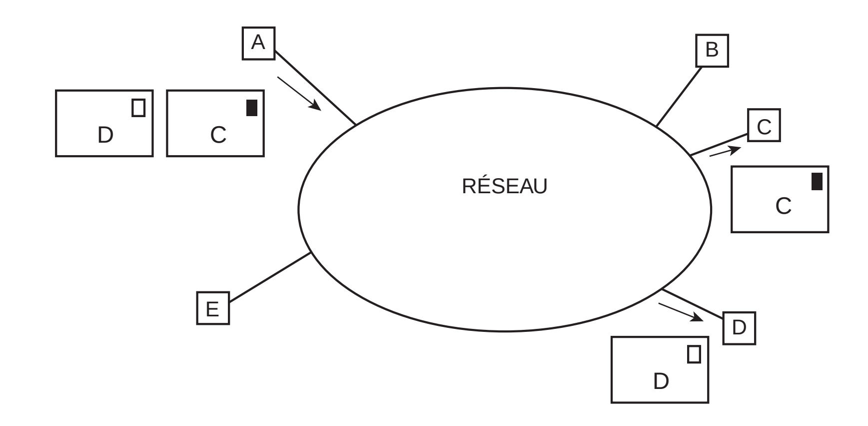


Figure 3.12

Service réseau sans connexion : un paquet peut être émis à tout moment, indépendamment des autres paquets et sans se soucier de l'état du destinataire.

1. D'autant que, de plus en plus, les adresses IP attribuées sont des adresses dynamiques, valables pour une durée déterminée (voir les détails au chapitre 6).



Dans un service en mode connecté ou orienté connexion, l'utilisateur doit d'abord indiquer avec qui il veut dialoguer. Pour cela, une procédure, appelée *ouverture de connexion*, établit un lien logique entre les deux équipements terminaux et constitue un « tube » de dialogue, appelé *circuit virtuel*. La connexion créée n'est active que si le destinataire accepte la communication. Ensuite, le réseau transmet tous les paquets de données jusqu'au destinataire, en se référant au circuit virtuel précédemment établi (l'émetteur n'a plus besoin de préciser l'adresse du destinataire dans chaque paquet). Lorsque le dialogue se termine, un des utilisateurs indique au réseau qu'il souhaite libérer la connexion. Pour dialoguer avec un autre équipement (ou le même), il faut déclencher une nouvelle ouverture de connexion. Le réseau téléphonique illustre un tel service : il faut décrocher le téléphone, composer le numéro de son correspondant, attendre qu'il réponde avant pouvoir dialoguer avec lui. Après avoir raccroché, il faut répéter les opérations précédentes si on veut communiquer à nouveau. La figure 3.13 montre une connexion établie entre les équipements terminaux *A* et *C*.

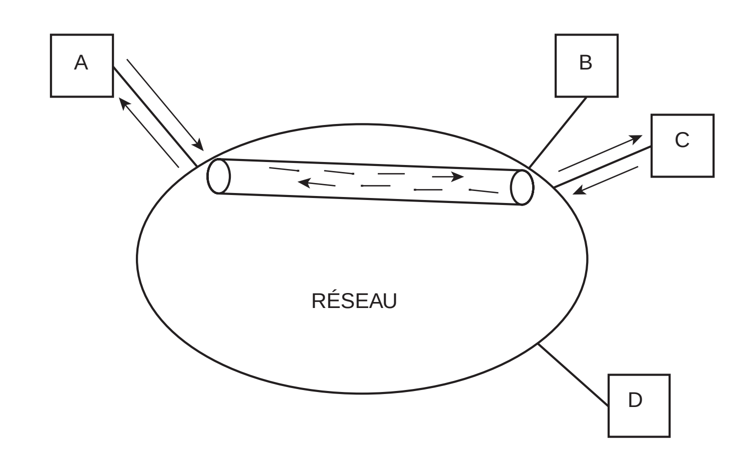


Figure 3.13

Service réseau en mode connecté entre les équipements A et C : un lien logique entre émetteur et récepteur est maintenu pendant toute la communication, mais les ressources physiques sont partagées.

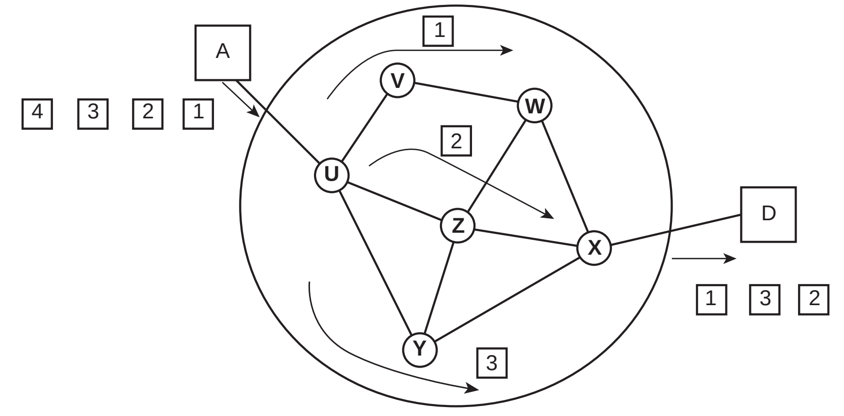
3.1 Service sans connexion

Un réseau à commutation de paquets qui offre un service sans connexion s'appelle couramment réseau à *datagrammes*, du nom des unités de données transportées. Un service sans connexion considère les différents datagrammes comme totalement indépendants les uns des autres. Chacun transite à travers le réseau avec l'ensemble des informations nécessaires à son acheminement. Il comprend notamment les adresses complètes de l'expéditeur et du destinataire. La fonction de routage s'exécute pour chaque datagramme. Ainsi, plusieurs datagrammes échangés entre les mêmes équipements terminaux peuvent suivre des chemins différents dans le réseau et le destinataire les recevoir dans un ordre différent de l'ordre d'émission. De plus, en cas de problème (rupture de liaison, manque de mémoire dans un commutateur), des datagrammes peuvent se perdre. L'équipement terminal doit non seulement reconstituer l'ordre des datagrammes reçus pour en exploiter correctement le contenu, mais aussi vérifier qu'aucun ne s'est égaré.

L'avantage d'un tel réseau est sa simplicité de réalisation interne : ce sont les équipements terminaux qui mettent en œuvre les fonctions de contrôle. La figure 3.14 montre l'acheminement des datagrammes entre les équipements *A* et *D*.

Figure 3.14

Acheminement des datagrammes entre les équipements A et D : l'ordre n'est pas garanti et il y a des pertes.



A envoie successivement les paquets 1, 2, 3, 4.

Le paquet 1 emprunte le chemin passant par les commutateurs U, V, W, X.

Les paquets 2 et 3 empruntent respectivement U, Z, X et U, Y, X. Le paquet 4 se perd.

D reçoit dans l'ordre 2, 3 puis 1 et ne reçoit pas 4.

3.2 Service avec connexion

Le service avec connexion est couplé avec la notion de circuit virtuel. À l'ouverture de la connexion, le réseau détermine le chemin que tous les paquets emprunteront par la suite. Ce chemin s'appelle « circuit virtuel ». Il s'agit d'un circuit car on utilise les mêmes principes que dans la commutation de circuits ; il est virtuel puisqu'une connexion ne monopolise une liaison entre commutateurs que pendant le temps de transfert d'un paquet. Une fois le paquet transmis, la liaison est utilisable par un autre circuit virtuel. La liaison entre deux commutateurs transporte donc plusieurs circuits virtuels entre des équipements terminaux totalement différents. De ce fait, l'utilisation du support de transmission est beaucoup plus efficace que dans le cas de la commutation de circuits.

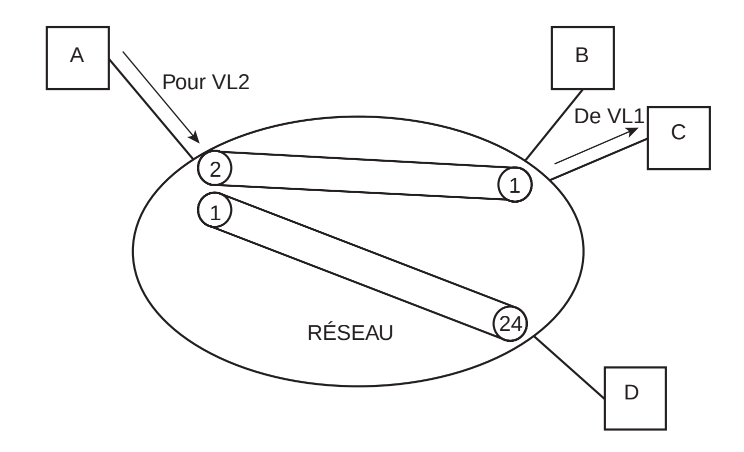
Un équipement terminal peut gérer plusieurs connexions en parallèle. Un identifiant, souvent appelé *numéro de voie logique*, les distingue. L'équipement émetteur précise l'adresse logique du destinataire à l'établissement d'une connexion. Il lui associe un numéro de voie logique. Le commutateur relié au récepteur attribue de son côté un numéro de voie logique à la future connexion. Les deux numéros de voie logique identifiant la connexion sont choisis indépendamment l'un de l'autre, pour la durée de la connexion. Ils constituent un adressage abrégé : les correspondants n'ont pas besoin de transporter dans leurs paquets les adresses complètes de l'émetteur et du destinataire. À titre de comparaison, lorsqu'un usager du téléphone affecte une touche du clavier à un numéro de téléphone particulier, il n'a pas à taper les 10 chiffres avant chaque appel.

L'équipement terminal place le numéro de voie logique approprié dans l'en-tête du paquet qu'il transmet. Celui-ci parvient au point d'accès du réseau et les commutateurs le propagent jusqu'au destinataire. Tous les paquets reçus et émis sur cette connexion portent donc

le même numéro de voie logique. La figure 3.15 montre un exemple de connexions multiples entre plusieurs équipements terminaux.

Figure 3.15

Exemple de connexions dans un réseau à commutation fonctionnant en mode connecté : signification locale des numéros de voie logique.



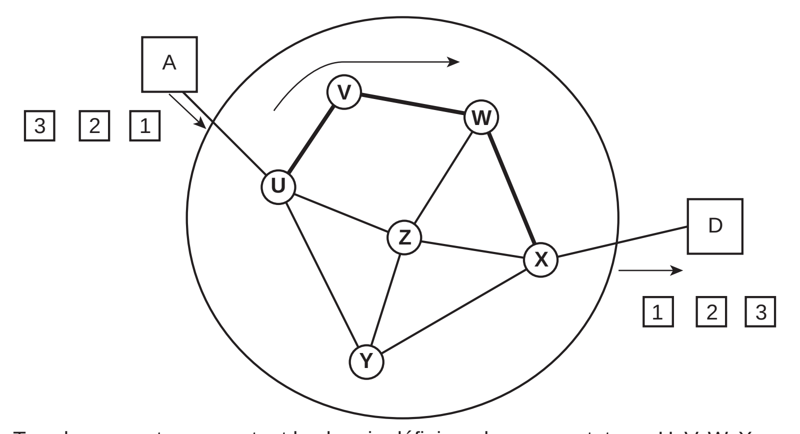
La voie logique 2 référence pour A sa connexion avec C, la voie logique 1 référence pour C sa connexion avec A. La voie logique 1 référence pour A sa connexion avec D, la voie logique 24 référence pour D sa connexion avec A. L'équipement A dispose de deux voies logiques 1 et 2 multiplexées sur la liaison avec le commutateur d'accès.

La correspondance entre l'adresse logique du destinataire (l'adresse complète de l'abonné) et le raccourci d'adressage qui l'identifie localement (le numéro de voie logique utilisé) est *bijective* (un numéro de voie logique n'identifie qu'un seul circuit virtuel, pour un échange de données bidirectionnel).

L'avantage d'un réseau à circuits virtuels (voir figure 3.16) est sa fiabilité : comme les paquets d'un même circuit virtuel suivent le même chemin, il suffit de conserver l'ordre des paquets sur chaque tronçon du chemin pour conserver globalement l'ordre des paquets sur le circuit virtuel. L'opérateur du réseau peut donc garantir une certaine qualité de service (taux d'erreur, contrôle de séquence et de flux…), au prix d'une plus grande complexité de réalisation et de gestion du réseau.

Figure 3.16

Exemple de réseau à circuits virtuels.



Tous les paquets empruntent le chemin défini par les commutateurs U, V, W, X.

4 Contrôles internes dans un réseau

Pour assurer le bon fonctionnement du réseau, l'opérateur ou l'administrateur du réseau exerce des fonctions de contrôle internes au réseau, principalement les fonctions de *routage*, de *contrôle de congestion* et d'*administration*. Présentons-les successivement.

4.1 Fonction de routage

Le routage détermine le chemin des paquets dans le réseau pour atteindre le correspondant désigné. Cette opération se fait à l'établissement d'un circuit virtuel ou réel et pour chaque datagramme dans un réseau à service sans connexion. Puisqu'il y a une multitude de chemins possibles dans un réseau maillé, le meilleur se choisit en fonction d'un critère qui peut être : le moins coûteux pour l'opérateur (ou pour le client), le plus rapide, le plus fiable…

Deux grandes catégories de fonctions existent : le routage *statique* et le routage *adaptatif*. Avec le premier, les nœuds du réseau choisissent à l'avance le chemin entre deux équipements et le mémorisent. Dans le second, le chemin varie en fonction de l'état du réseau et tient compte des pannes (de liaisons ou de commutateurs) ou du trafic écoulé par le réseau. Ce type de routage utilise au mieux les ressources du réseau et améliore sa défense en cas d'incident. En outre, un algorithme de routage peut être *local*, *réparti* dans tout le réseau, ou *centralisé*.

Dans un routage local, chaque commutateur détermine le chemin vers le destinataire sur la base d'informations locales : la taille de ses files d'attente, l'occupation des lignes qui le raccordent aux autres commutateurs… Il n'a pas connaissance de l'environnement, c'est-à-dire de l'état des commutateurs voisins.

Un exemple de routage local est le routage par « inondation » : un commutateur envoie tout paquet reçu sur toutes les liaisons, hormis celle d'où il provient (voir figure 3.17). Cet algorithme provoque une multiplication, en théorie infinie, du nombre de paquets. Il faut donc en détruire certains pour éviter la congestion du réseau. Pour cela on place, dans l'en-tête de chaque paquet, un compteur que les commutateurs décrémentent à chaque envoi. Lorsque la valeur du compteur s'annule, le paquet est détruit. La valeur minimale initiale doit être égale au nombre minimal de commutateurs à traverser pour atteindre le correspondant. Si le nombre de sauts est inconnu de l'expéditeur, on peut prendre le nombre maximal de commutateurs séparant deux équipements quelconques. Les avantages principaux du routage par inondation sont la simplicité et la fiabilité, puisqu'il trouve toujours le chemin le plus court (lorsqu'il existe), quel que soit l'état du réseau. Les militaires l'utilisent, le fonctionnement de leur réseau (transmission d'alarmes) devant être assuré coûte que coûte, même si un grand nombre de commutateurs ou de liaisons est détruit.

Un autre algorithme de routage très simple est celui de la « patate chaude » (*hot potatoe*), qui consiste pour un commutateur à se débarrasser le plus rapidement possible d'un paquet reçu, en le transmettant sur la liaison la moins chargée (hormis celle d'où vient ce paquet). Un tel algorithme est adaptatif puisqu'il prend en compte l'état du réseau. Pour améliorer son efficacité, il peut être combiné avec un routage statique qui mémorise plusieurs routes possibles.

Dans le cas d'un routage centralisé, un équipement spécialisé est dédié à la fonction de calcul de toutes les routes. L'ensemble des commutateurs interroge cet équipement à chaque opération de routage. Pour un routage adaptatif, l'équipement exécutant l'algorithme de routage doit connaître en permanence l'état complet du réseau. Dans un routage



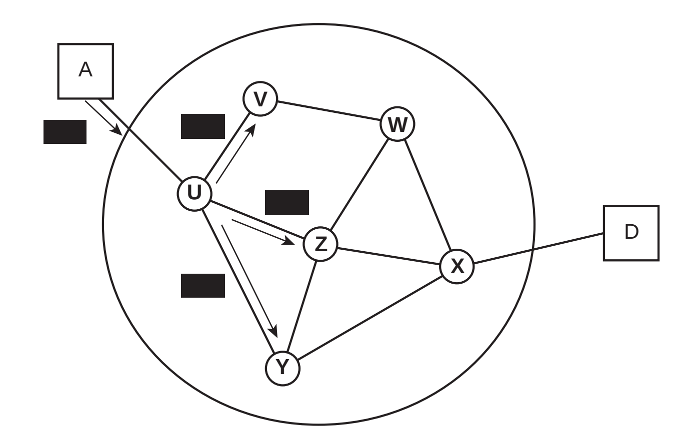


Figure 3.17

Routage par inondation dans un réseau : le paquet suit toutes les directions possibles.

réparti, chaque nœud diffuse à ses voisins des indications sur son état. Un nœud peut donc déterminer un chemin en fonction de son propre état et de l'état de ses proches voisins. La description de l'ensemble des algorithmes de routage sort du cadre de ce livre. Ils sont variés et peuvent être assez complexes. Nous nous sommes contentés ici de donner deux exemples simples de routage dans un réseau à commutation de paquets : le routage par inondation et celui de la patate chaude.

4.2 Contrôle de congestion

Le contrôle de congestion est l'ensemble des opérations qu'il faut effectuer pour éviter que les ressources des commutateurs soient saturées. L'efficacité de la fonction de routage est, à ce titre, fondamentale car elle doit répartir le trafic entre les commutateurs.

On peut mentionner plusieurs méthodes : perte délibérée de paquets, limitation du nombre de connexions et le contrôle isarythmique. Perdre des paquets est une méthode radicale qui vide la mémoire d'un commutateur ! Celui-ci récupère des ressources pour la suite. En général, on compte sur le fait que tous les paquets jetés et qui vont être retransmis par les utilisateurs ne le seront pas tous en même temps. La limitation du nombre des connexions consiste à refuser de nouvelles connexions si le niveau de disponibilité des ressources dépasse un certain seuil. Le contrôle isarythmique peut se substituer au contrôle du nombre des connexions ou s'y ajouter : il consiste à maîtriser le nombre de paquets entrant dans le réseau : chaque commutateur d'accès dispose d'un certain nombre de « crédits », tout paquet entrant consomme un crédit et tout paquet sortant du réseau libère un crédit. Lorsque le commutateur n'a plus de crédits, il bloque les paquets à l'entrée. La description précédente n'est pas exhaustive car il existe de nombreuses méthodes de contrôle de congestion, qui peuvent être combinées.

4.3 Administration des réseaux

Administrer un réseau revient à gérer au mieux sa mise en œuvre opérationnelle. Or, les architectures actuelles ne sont pas homogènes car il n'existe pas de système permettant de répondre à l'ensemble des besoins d'un utilisateur.

De plus, la gestion du réseau ne se limite pas à la bonne gestion du service de transport de l'information. Elle implique également la gestion correcte de son traitement. L'utilisateur a donc besoin d'une gestion puissante, qui tienne compte de l'hétérogénéité de l'architecture du réseau et lui fournisse un véritable « système d'exploitation réseau » prenant en

charge les aspects distribués du système. Les besoins, en matière de gestion, se situent donc à deux niveaux : celui de l'*utilisateur* et celui de l'*opérateur* du réseau. Présentons-les successivement.

Besoins de l'utilisateur

Les besoins de l'utilisateur sont très variés et s'expriment par l'accès aux applications et aux serveurs, la confidentialité des échanges, l'assistance technique et la qualité du service.

L'utilisateur demande tout d'abord de pouvoir se connecter aux différentes applications, grâce à un ensemble d'outils d'accès. Il a besoin, dans certains cas, d'accéder aux serveurs de noms afin de localiser une ressource. Il peut souhaiter que ses échanges soient confidentiels (inexploitables pour celui qui les aurait indûment récupérés). Depuis sa console, il n'a *a priori* aucune connaissance de l'architecture du système sur lequel il est connecté. À cause de pannes ou pour des raisons personnelles, il pourrait avoir besoin de conseils pour se sortir d'une situation anormale ou inconnue. Il apprécie alors les aides et les modes opératoires qui lui sont fournis pour surmonter ces cas d'exception.

Enfin, la qualité de service est un élément important dans la gestion d'un réseau car elle est directement ressentie par l'utilisateur. Elle correspond aux notions de disponibilité du système (pourcentage du temps pendant lequel le système fonctionne) et de performances attendues (temps de réponse, taux de perte…).

Besoins de l'opérateur

Les besoins de l'opérateur sont également variés et se déclinent de multiples façons : planification, exploitation et maintenance, prise en compte de l'hétérogénéité, résistance aux pannes, connaissance des chaînes de liaison.

L'opérateur d'un réseau recherche constamment l'adéquation entre les ressources de son système et les besoins de ses utilisateurs. La planification doit harmoniser l'ensemble des ressources disponibles par rapport aux demandes, tout en optimisant les coûts. Il doit donc disposer d'une vue globale du système, pour effectuer la répartition des ressources offertes et suivre l'évolution du système dans le temps.

La phase d'exploitation du réseau correspond au suivi permanent des ressources. Elle représente l'ensemble des actions journalières menées par une équipe réseau. L'exploitation s'effectue par la surveillance des différents composants constituant le réseau. C'est, en général, lors de cette phase qu'on détecte les anomalies (logicielles ou matérielles) de fonctionnement auxquelles il faut remédier. L'opération est plus ou moins aisée, selon les outils disponibles (outils de tests logiciels aussi bien que matériels). Le but est de réparer au plus vite les éléments défectueux. En général, le service d'exploitation intervient pour localiser au mieux la cause de l'anomalie et propose, si possible, une solution de secours. Après réparation par l'équipe de maintenance, l'exploitation réintègre l'ensemble des composants dans le réseau.

La prise en compte de l'hétérogénéité d'un réseau est un véritable problème pour le gestionnaire. En effet, il faut pouvoir corréler l'ensemble des états des systèmes, afin d'établir des relations de cause à effet. Il faut aussi mettre en évidence des situations de fonctionnement anormal et analyser finement les événements qui y ont conduit. La complexité du problème provient du fait que les différents éléments constituant l'architecture d'un réseau ne fonctionnent pas nécessairement suivant les mêmes normes : ils ne fournissent donc pas des informations directement comparables.

La qualité d'une gestion se mesure au fait qu'elle est capable, dans tous les cas, de continuer sa surveillance et donc de résister aux pannes. Cela implique qu'une panne isolée ne



puisse invalider la gestion. En effet, il serait dérisoire d'implanter une application qui bloque le système, surtout si le but de cette application est de fournir des remèdes à ces blocages !

La connaissance des chaînes de liaison (c'est-à-dire la succession des équipements matériels et logiciels qui interviennent dans une communication) est importante. Elle fournit un suivi dynamique des différents échanges survenant dans le système distribué. On peut ainsi suivre une ou plusieurs communications. Cette connaissance permet, entre autres, de rétablir des connexions et de masquer à l'utilisateur le chemin d'accès à son application.

Résumé

Une infrastructure de communication optimise les coûts de fonctionnement et de maintenance d'un réseau reliant un grand nombre d'équipements informatiques. Elle utilise différentes techniques de commutation pour organiser le partage des ressources. La commutation de circuits est la technique employée dans le réseau téléphonique alors que la commutation de paquets sert dans les échanges de données informatiques. Deux services réseau peuvent s'utiliser pour transférer les données. Le premier service est évolué, il est normalisé sur le plan international et fonctionne en mode connecté. Il est couramment appelé circuit virtuel et transporte des paquets en garantissant leur séquence et leur intégrité. L'autre, plus simple, s'utilise à l'échelle mondiale dans Internet. Il fonctionne en mode non connecté et transfère des datagrammes indépendants les uns des autres sans leur apporter de contrôle. Enfin, des fonctions de contrôle interne – routage, contrôle de congestion et administration – assurent la bonne marche d'un réseau.

Problèmes et exercices

Exercice 1 Choix d'un service réseau

Énoncé

Un message de 40 octets doit être transmis entre deux équipements *A* et *B*. Supposons qu'on puisse connecter ces deux équipements à trois types de réseaux : (1) un réseau à commutation de circuits, (2) un réseau à commutation de paquets offrant un service orienté connexion, (3) un réseau à commutation de paquets offrant un service sans connexion. Quel type de réseau choisiriez-vous pour réaliser ce transfert de données ?

Solution

Sans autres précisions sur la nature du message et du degré de fiabilité souhaitée pour le transfert, il serait beaucoup plus simple d'utiliser un réseau à commutation de paquets, offrant un service sans connexion. Cela évite d'établir la connexion puis de la libérer après transfert (étapes obligatoires dans les réseaux en mode connecté) pour une si petite taille de message.

Exercice 2 Affectation des numéros de voie logique aux circuits virtuels

Énoncé

Deux entités *A* et *B* communiquent à travers un réseau de données utilisant des circuits virtuels. Pour communiquer, ils ont établi un circuit virtuel, identifié par le numéro de voie logique 152 du côté de *A*. Peut-on en déduire le numéro de voie logique utilisé par *B* ?

Solution

Non, il est impossible de déduire le numéro de voie logique utilisé du côté de *B* à partir du numéro employé par *A*. En effet, les numéros de voie logique sont définis localement, au niveau de l'interface entre l'équipement terminal et le nœud d'accès au réseau. Les deux numéros sont donc choisis indépendamment l'un de l'autre. Il n'existe aucune corrélation entre eux.

Remarque

Les numéros de voie logique ne sont pas choisis au hasard par les équipements. Si tel était le cas, un même numéro pourrait être attribué trop souvent à deux communications différentes, créant une *collision d'appels*. Pour éviter cela, l'équipement appelant (celui qui demande l'ouverture de connexion) choisit le plus grand numéro de voie logique disponible localement, alors que l'équipement appelé se verra affecté du plus petit numéro de voie logique disponible sur l'interface locale (en général à partir de 1, le numéro 0 étant réservé à l'administration du réseau). Le risque de collision d'appels est alors minimal.

EXERCICE 3 OUVERTURE DE PLUSIEURS CIRCUITS VIRTUELS ENTRE ÉQUIPEMENTS TERMINAUX

Énoncé

On considère un réseau à commutation de paquets offrant un service orienté connexion reliant plusieurs ETTD.

a Peut-on ouvrir simultanément plusieurs circuits virtuels entre deux ETTD ? Combien ? De quoi dépend ce nombre ?

b Si vous avez répondu par l'affirmative dans la précédente question, quel serait l'intérêt d'une telle solution ? Sinon, quel est l'inconvénient ?

Solution

a *A priori*, seul l'ETTD choisit avec qui il veut établir un circuit virtuel. Il peut naturellement en ouvrir plusieurs avec le même correspondant s'il le désire. Le nombre de circuits virtuels qu'il peut ouvrir simultanément dépend de deux facteurs : l'abonnement contracté et la taille du champ dévolu à l'identification du numéro de voie logique. Si celle-ci est, par exemple, 12 bits, on pourra identifier localement 4 096 circuits virtuels différents. Le nombre de circuits virtuels utilisables simultanément est un service facturé à l'utilisateur.

Remarque

La norme décrivant l'utilisation des circuits virtuels spécifie la taille du champ définissant le numéro de voie logique. Les réseaux publics utilisant les circuits virtuels s'appuient sur la recommandation *X.25* de l'ITU. Cette norme définit trois niveaux : le type d'interface série à utiliser, le protocole de liaisons de données à employer (généralement LAP-B, c'est-à-dire le mode équilibré de HDLC avec rejet simple des erreurs) et le mode d'acheminement des données (les paquets sont émis sur des circuits virtuels). Dans *X.25*, la taille du champ affecté aux numéros de voie logique est 12 bits.

b L'intérêt d'utiliser plusieurs circuits virtuels entre les deux mêmes ETTD est d'augmenter le flux des données entre eux, puisqu'à chaque circuit virtuel est associé un débit binaire maximal. Le débit binaire entre les deux ETTD est presque multiplié[2] par le nombre de circuits virtuels affectés à cette communication. Il faut toutefois disposer, aux deux extrémités, d'outils capables de réordonnancer les paquets provenant des différents circuits virtuels. Supposons qu'un ETTD utilise trois circuits virtuels différents pour communiquer avec le même ETTD distant. Il envoie ses données en utilisant un mécanisme d'« éclatement » des données sur les trois circuits virtuels (par exemple, en émettant le premier paquet sur le premier circuit virtuel, le deuxième paquet sur le deuxième circuit virtuel et ainsi de suite). L'ordre des paquets sera respecté au sein de chaque circuit virtuel mais ne correspondra pas forcément à l'ordre initial des paquets du message.

Remarque

Le débit binaire maximal disponible est lui aussi un paramètre négocié à l'abonnement. Il est complété par la notion de *classe de service*, qui garantit à l'utilisateur un débit binaire minimal pour la durée de la connexion. Par exemple, un abonné peut posséder un accès à 56 kbit/s et demander une classe de service de 9 600 bit/s. Cela signifie que, quel que soit l'encombrement du réseau, le débit binaire de ce circuit virtuel ne pourra être inférieur à cette valeur.

2. La transmission des paquets sur plusieurs circuits virtuels nécessite divers traitements au niveau des équipements terminaux. Le débit global est donc inférieur à la somme des débits des différents circuits virtuels utilisés pour le transfert des données.

EXERCICE 4 MULTIPLEXAGE DE CIRCUITS VIRTUELS SUR LA MÊME LIAISON DE DONNÉES

Énoncé

Un ETTD *A* établit 4 connexions avec 4 autres ETTD dénommés *B*, *C*, *D* et *E*. Il utilise pour cela un circuit virtuel par ETTD distant. Les numéros de voie logique affectés aux circuits virtuels valent respectivement 154, 153, 152 et 151. On suppose que *A* doit envoyer une information qui tient en 4 paquets à chacun des 4 destinataires.

a En supposant que les ETTD contactés n'aient aucune communication en cours, quel(s) numéro(s) de voie logique utiliseront-ils pour la communication avec l'ETTD *A* ?

b Si un paquet est contenu dans une seule trame I, combien de trames transportant les paquets à destination des ETTD distants seront émises par *A* ? Combien les autres ETTD recevront-ils de trames ?

c Comment s'effectue la répartition des paquets de données dans les différentes trames I au niveau de *A* ?

Solution

a Chaque ETTD distant n'ayant aucune communication en cours, il utilisera le numéro de voie logique 1 pour identifier le circuit virtuel avec l'ETTD *A*. Cela est dû au fait que l'attribution des numéros de voie logique est locale à l'interface ETTD – nœud d'accès au réseau.

b Il y a donc 16 trames I émises par l'ETTD *A*, chacune contenant un paquet à destination de l'un des ETTD distants. Par contre, chaque ETTD distant ne reçoit que 4 trames, contenant les 4 paquets qui lui sont destinés.

c Chaque ETTD distant reçoit ses paquets dans l'ordre où ils ont été émis par *A*. On ne peut pas en déduire pour autant l'ordre des trames I qui sont émises vers le nœud d'accès de *A*. En effet, chaque paquet à émettre est placé dans la file d'attente des paquets affectée au circuit virtuel concerné ; il y a autant de files d'attente que de circuits virtuels actifs.

Ensuite, l'entité qui gère la liaison de données prélève les données à émettre dans l'une des files d'attente, selon un ordre qui lui est propre (et qui dépend de la conception du logiciel). Ainsi, le gestionnaire de liaison peut décider d'entrelacer les données à destination des différents circuits virtuels (il envoie le premier paquet du premier circuit puis le premier paquet du deuxième circuit, et ainsi de suite), ou bien il peut envoyer tous les paquets d'un même circuit virtuel avant d'envoyer ceux du circuit virtuel suivant. Au moment d'émettre la trame I, le paquet remplit le champ de données de la trame.

Remarque

Cet exercice montre comment les entités du niveau immédiatement inférieur rendent les services à une entité de niveau supérieur. Il met aussi en évidence l'indépendance des entités des différents niveaux. En effet, le niveau gérant les paquets contient une ou plusieurs entités gérant les circuits virtuels, conformément au protocole de gestion des paquets. Chaque entité de ce niveau va soumettre des paquets à l'entité gérant la couche Liaison de données. Cette dernière est donc la seule à décider du mode d'envoi des données sur la liaison, en fonction de l'activité des différents circuits virtuels.



EXERCICE 5 CALCUL DU TEMPS DE TRANSMISSION DANS UN RÉSEAU À COMMUTATION

Énoncé

Soit un réseau à commutation au sein duquel deux stations *A* et *B* ont établi une communication. *A* doit envoyer un fichier de taille *L* bits à *B*. Le transfert de données présente les caractéristiques suivantes :

- *S* est le nombre de commutateurs traversés pour la communication entre *A* et *B*.
- Toutes les liaisons de données utilisées ont un débit *D* bit/s.
- Le protocole de liaison est le même sur toutes les liaisons ; il ajoute un en-tête de *H* bits à chaque unité de données transférée.
- On néglige les temps de propagation et les temps de traitement dans les commutateurs du réseau. On néglige de même les temps de gestion des accusés de réception.

a Le réseau est un réseau à commutation de messages. Le fichier est transmis dans un seul message, d'une liaison à l'autre, jusqu'au destinataire. Donnez l'expression T_{fic1} du temps de transmission de ce fichier dans le réseau.

b Le réseau est un réseau à commutation de paquets. Le fichier est découpé en paquets contenant *P* bits de données (pour simplifier, on supposera que les paquets sont tous de taille identique). Montrez que l'expression T_{fic2} du temps de transmission du fichier est : $T_{fic2} = (S + L/P)(P + H)/D$.

c Calculez et comparez les temps obtenus dans les deux premières questions en prenant : $L = 64\,000$ octets ; $H = 9$ octets ; $S = 2$; $D = 64$ kbit/s. On prendra trois valeurs possibles pour la taille des paquets : $P = 128$ octets ; $P = 16$ octets ; $P = 48$ octets (dans ce dernier cas, il s'agit d'une cellule ATM dont l'en-tête *H* utilise 5 octets).

d Quels sont les avantages et les inconvénients de la commutation de paquets par rapport à la commutation de messages ?

e Les liaisons sont affectées d'un taux d'erreurs noté τ. Montrez que la probabilité *p* pour qu'une trame de longueur *L* soit reçue correctement vaut $p = (1 - \tau)^L$. En déduire que le nombre moyen *N* de transmissions d'une trame vaut : $N = 1/p$. Pour obtenir ce résultat, on supposera que le protocole de liaison répète indéfiniment la même trame sans anticipation, jusqu'à ce que la trame soit correctement reçue.

f Refaire l'application numérique de la question c en prenant un taux d'erreurs $\tau = 10^{-4}$. Pour ces calculs, on considère qu'une seule trame est émise dans le réseau à commutation de messages. Dans la commutation de paquets, chaque paquet est transmis dans une trame.

g Comparez les résultats et concluez. Ces techniques sont-elles adaptées aux hauts débits ?

Solution

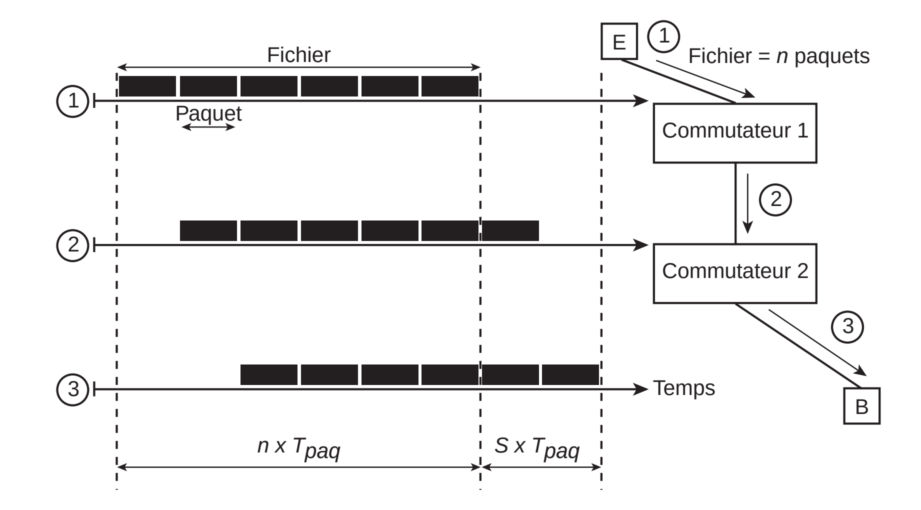
a La durée de transmission du fichier sur une liaison est égale à : $T_{fic} = (L + H)/D$. La durée de transmission du fichier est égale au temps de transmission sur toutes les liaisons traversées, c'est-à-dire : $T_{fic1} = T_{fic}*(S + 1)$.

b La durée de transmission d'un paquet sur une liaison de données vaut : $T_{paq} = (P + H)/D$. La durée de transmission du fichier est égale à la durée de transmission des paquets jusqu'au premier commutateur, plus le délai nécessaire au dernier paquet pour parvenir jusqu'à *B*. Le nombre de paquets nécessaires pour transmettre le fichier vaut $n = L/P$. On en déduit : $T_{fic2} = (S + n)*T_{paq} = (S + n)*(P + H)/D$.

La figure 3.18 montre comment calculer les différents temps de transmission.

Figure 3.18

Calcul des différents temps de transmission.



c Applications numériques :

a. Cas de la commutation de messages : $P = L = 64\,000*8 = 512\,000$ bits

$$T_{fic1} = (2 + 1)*(64\,000 + 9)*8/64\,000 = 24 \text{ s.}$$

b. Cas de la commutation de paquets avec $P = 128$ octets = 128*8 = 1 024 bits ; $n = L/P = 500$ paquets

$$T_{fic2} = (2 + 500)*(128 + 9)*8/64\,000 = 8{,}6 \text{ s.}$$

c. Cas de la commutation de paquets avec $P = 16$ octets = 16*8 = 128 bits ; $n = L/P = 4\,000$ paquets

$$T_{fic2} = (2 + 4\,000)*(16 + 9)*8/64\,000 = 12{,}5 \text{ s.}$$

d. Cas de la commutation de cellules ATM avec $P = 48$ octets = 48*8 = 384 bits ; $H = 5$ octets ; $n = L/P = 1\,334$ paquets (par excès)

$$T_{fic2} = (2 + 1\,334)*(48 + 5)*8/64\,000 = 8{,}85 \text{ s.}$$

Remarque

Pour la commutation de messages, le temps de transmission du fichier ne dépend que du nombre de liaisons traversées. En revanche, pour la commutation de paquets, il faut tenir compte du recouvrement des temps de transmission des différents paquets sur l'ensemble des liaisons : en effet, pendant que *A* transmet son deuxième paquet au premier commutateur, celui-ci envoie le premier paquet au commutateur suivant et ainsi de suite. C'est la raison pour laquelle les performances des réseaux à commutation de paquets sont supérieures à celles des réseaux à commutation de messages. L'écart des performances sera encore plus notable si certaines liaisons transmettent le message avec des erreurs, comme nous le verrons aux questions suivantes.

d Nous voyons bien que le découpage en paquets permet de réduire les délais d'acheminement à travers le réseau. Cependant, il faut respecter une juste proportion entre la taille de l'en-tête par rapport au corps du message : une taille de paquet trop petite provoque un allongement de délai d'acheminement.

e Pour qu'une trame de longueur L soit reçue sans erreur, il faut que tous ses bits soient reçus sans erreur. La probabilité de recevoir un bit sans erreur vaut $1 - \tau$. La probabilité de recevoir L bits sans erreur vaut : $(1 - \tau)^L$. La probabilité de recevoir une trame erronée est de $p_t = 1 - (1 - \tau)^L$.

Puisque la longueur d'une trame vaut $L = P + H$, le nombre moyen d'émissions est donc :

$$1^*(1 - p_t) + 2^*(1 - p_t)\, p_t + 3^*(1 - p_t)\, p_t^2 + \ldots = 1/(1 - p_t).$$

En appliquant la formule précédente en tenant compte des répétitions, on obtient :

$$T'_{fic} = T_{fic}/(1 - p_t) = T_{fic}/1 - (1 - \tau)^L.$$

f Les applications numériques donnent :

a. Cas de la commutation de messages : $P = L = 64\,000^*8 = 512\,000$ bits

$T'_{fic} = 16\,848$ s soit plus de 4 heures !

b. Cas de la commutation de paquets avec $P = 128$ octets = 128*8 = 1 024 bits

$T'_{fic} = 9{,}6\ s$, soit une dégradation de 11,6 % par rapport au cas parfait.

c. Cas de la commutation de paquets avec $P = 16$ octets = 16*8 = 128 bits ;
$n = L/P = 4\,000$ paquets

$T'_{fic} = 12{,}75\ s$, soit une dégradation de 2 % par rapport au cas parfait.

$T'_{fic} = 9{,}22\ s$, soit une dégradation de 4,2 % par rapport au cas parfait.

g La prise en compte du taux d'erreurs dans les liaisons montre tout l'intérêt du découpage des messages en paquets. Il est visiblement hors de question d'utiliser la commutation de messages pour les applications nécessitant les hauts débits, tout particulièrement lorsque les liaisons sont peu fiables. Nous voyons également qu'une taille de paquet trop petite est un choix peu judicieux. Les cellules ATM et les paquets de 128 octets sont donc des compromis intéressants entre les différentes contraintes pour les hauts débits.

EXERCICE 6 TRANSFERT DE DONNÉES SUR CIRCUIT VIRTUEL

Énoncé

Après avoir fait vos courses dans l'hypermarché proche de votre domicile, vous vous présentez à une caisse pour régler vos achats. Vous décidez de payer avec la carte du magasin. Examinons le déroulement des opérations entre votre caisse et le centre de traitement informatique de la chaîne de magasins *via* Transpac, le réseau public de données français utilisant le protocole *X.25*.

La caisse C de l'hypermarché possède un accès avec les caractéristiques suivantes : le débit binaire est de 48 000 bit/s ; le niveau Liaison de données utilise le mode équilibré du protocole HDLC (protocole LAP-B) avec une fenêtre d'anticipation de 4. La gestion des paquets de données est faite sur un CVC (circuit virtuel commuté), ouvert par la caisse dès sa mise en service (le CVC reste ouvert jusqu'à la fermeture du magasin : inutile de le fermer après chaque opération de la caisse). La taille de la fenêtre d'anticipation des paquets est 1.

Le serveur S de la chaîne d'hypermarchés possède un accès configuré comme suit : le débit binaire est de 2,048 Mbit/s ; le niveau Liaison de données utilise le même protocole LAP-B avec une fenêtre d'anticipation de 7 ; le CVC possède une fenêtre d'anticipation des paquets de 1.

☞



Après avoir saisi le prix de tous les articles que vous avez achetés, la caissière appuie sur une touche qui provoque l'envoi d'une requête *Demande de paiement par carte magasin*. Cette requête parvient au centre de traitement de la chaîne de magasins et active l'application de facturation. Celle-ci traite la transaction et renvoie une *Autorisation de paiement par carte magasin*. La caisse affiche alors sur l'écran un message vous invitant à taper le chiffre qui précise la nature de votre paiement (à comptant ou à crédit), puis les chiffres de votre code secret. Vous pianotez votre choix de paiement puis votre code sur le pavé numérique. La caissière vérifie que la saisie est correcte et valide les données fournies en appuyant sur une touche. La caisse transmet les données que vous avez tapées dans un paquet *Données du paiement EFT*[3], puis elle reçoit un paquet *Fin de transfert EFT*, qui provoque l'affichage du message d'accueil du client suivant sur l'écran de la caisse.

a En supposant que le plus grand numéro de voie logique disponible soit 9 et le plus petit soit égal à 1 pour les deux ETTD, décrivez l'échange des paquets nécessaires à l'ouverture du CVC par la caisse, en respectant les conventions d'attribution des numéros de voie logique. Pour ouvrir le CVC et transférer les données, vous disposez des types de paquets donnés au tableau 3.1.

b En supposant que chaque message ou requête tienne dans un seul paquet de données, décrivez le transfert des paquets de données entre la caisse et son ETCD (le nœud d'accès au réseau) d'une part, et entre le centre de traitement et son ETCD d'autre part. Vous veillerez à respecter la chronologie des événements (par exemple, un paquet émis par le serveur en réponse à un paquet de la caisse doit être émis *après* que le paquet de la caisse est parvenu au serveur…).

c Pourquoi le serveur *S* utiliserait-il le numéro de voie logique 5 pour identifier la connexion avec la caisse *C* ?

d Indiquez comment s'effectue la fermeture du CVC par la caisse à la fin de la journée.

e Si des parasites sur les liaisons de données avaient produit des erreurs de transmission, quel niveau les détecterait ?

Tableau 3.1 : Les différents types de paquets utilisés sur un circuit virtuel

Type du paquet	Fonction	Paramètres	Mnémonique utilisé
Ouverture de connexion	Ouvre le circuit virtuel	O, D, n° VL	APPEL (O, D, n° VL) (a)
Indication d'ouverture	Alerte l'ETTD distant	O, D, n° VL	AP_ENT (O, D, n° VL) (b)
Acceptation d'ouverture	Accepte la communication	O, D, n° VL	COM_ACC (O, D, n° VL) (c)
Indication d'acceptation	Indique à l'ETTD appelant que l'appelé accepte la connexion	O, D, n° VL	COM_ETA (O, D, n° VL) (d)
Demande de libération	Un ETTD demande la fin du transfert de données	O, D, n° VL	DEM_LIB (O, D, n° VL) (e)

☞

3. EFT signifie *Electronic Fund Transfer* (transfert de fonds électronique).

Énoncé (suite)

Tableau 3.1 : Les différents types de paquets utilisés sur un circuit virtuel (suite)

Type du paquet	Fonction	Paramètres	Mnémonique utilisé
Indication de libération	Indique une demande de fin de transfert de l'autre ETTD	O, D, n° VL	IND_LIB (O, D, n° VL) (f)
Confirmation de libération	Confirme la fin du transfert de données	O, D, n° VL	LIB_CONF (O, D, n° VL)
Acquittement des données	Acquitte les paquets de données jusqu'à *P(R) – 1*	X	PRR(X) (g)
Paquet de données	Numérote le paquet émis et acquitte les paquets reçus par piggy-backing	X,Y, Z	DXY(Z) (h)

Au tableau 3.1, nous utilisons les notations suivantes :

(a) *O* est l'adresse complète de l'abonné origine ; *D* l'adresse complète de l'abonné distant ; *n° VL* est le numéro de voie logique localement utilisé pour identifier le circuit virtuel à ouvrir.

(b) *AP_ENT* est la notation abrégée utilisée pour un appel entrant. Ce paquet est émis par le nœud de l'ETTD appelé pour l'avertir d'une demande de connexion par un ETTD distant.

(c) *COM_ACC* signifie que l'ETTD distant accepte la demande de connexion. Ce paquet est émis par l'ETTD appelé vers son ETCD.

(d) *COM_ETA* sert à indiquer à l'ETTD appelant que l'ETTD distant a accepté la communication. Ce paquet est émis par le nœud de l'ETTD origine.

(e) *DEM_LIB* signifie que l'ETTD qui l'émet demande la fin du transfert de données.

(f) *LIB_CONF* est la confirmation de la libération demandée soit par le réseau (en cas de panne ou d'impossibilité de traiter l'appel), soit par l'ETTD distant.

(g) *PRR(X)* est le paquet d'acquittement des paquets reçus en séquence jusqu'au paquet de numéro *P(R)* – 1.

(h) *DXY(Z)* est un paquet de données dans lequel *X* représente le numéro d'ordre *P(S)*, *Y* est l'acquittement des paquets jusqu'à *P(R) – 1* et *Z* le numéro de voie logique utilisé.

Solution **a** La figure 3.19 montre l'échange des paquets entre les ETTD et les ETCD pour ouvrir le CVC. *C* utilise le plus grand numéro de voie logique et *S* le plus petit disponible sur l'interface locale.

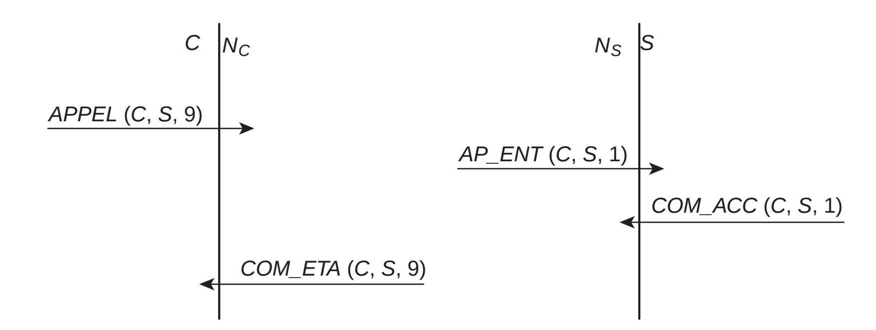
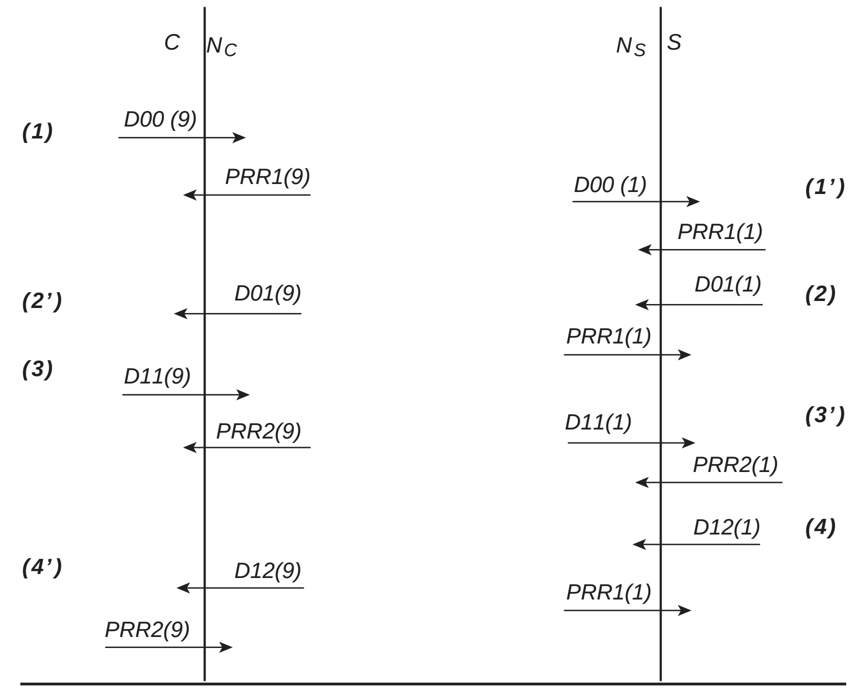


Figure 3.19
Ouverture du CVC par la caisse.

b La figure 3.20 nous montre les échanges de paquets entre ETCD et ETTD pour assurer le transfert de données demandé.

Figure 3.20

Transfert de données entre la caisse et le serveur.



(1) : Paquet contenant « Demande de paiement par carte magasin ».

(1') : Le paquet contenant « Demande de paiement par carte magasin » est arrivé dans le nœud du serveur. Il est envoyé au serveur et va servir à lancer la transaction de paiement.

(2) : Paquet contenant « Autorisation de paiement par carte magasin », généré par l'application de facturation, après vérification des données du client. Ce paquet est envoyé au nœud du serveur.

(2') : Le paquet contenant « Autorisation de paiement par carte magasin » est arrivé au nœud de la caisse, qui le transmet à la caisse. Il va servir à afficher le message vous invitant à taper votre code secret.

(3) : Paquet contenant « Données du paiement EFT », contenant vos données client.

(3') : Le paquet contenant « Données du paiement EFT » est arrivé au nœud du serveur, qui le retransmet au serveur. Il va permettre de mettre à jour votre compte client dans le serveur.

(4) : Paquet contenant « Fin de transfert EFT » pour demander la fin de la transaction.

(4') : Le paquet contenant « Fin de transfert EFT » arrive au nœud de la caisse. Ce dernier le retransmet à la caisse. Il va permettre d'afficher le message d'accueil du client suivant.

c Dans la réponse à la question précédente, nous supposons implicitement que le serveur n'a pas d'autres circuits virtuels ouverts quand il reçoit la demande de connexion de la caisse *C*. S'il utilise maintenant le numéro de voie logique 5, c'est tout simplement parce qu'il a déjà utilisé les quatre premiers numéros de voie logique pour d'autres connexions.

d La figure 3.21 montre la fermeture du CVC par la caisse.



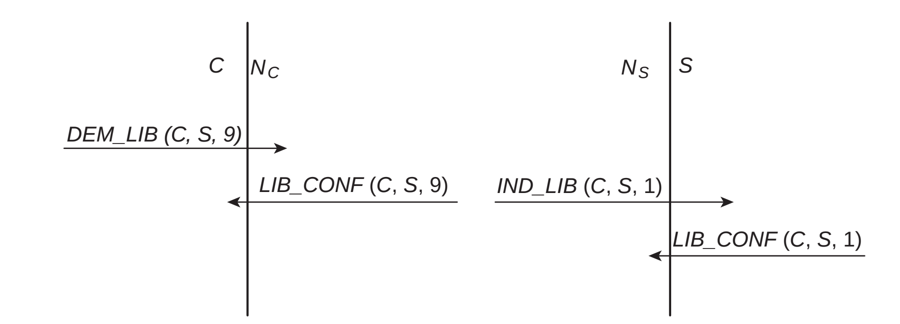


Figure 3.21

Fermeture du CVC par la caisse.

> **Remarque**
>
> Nous avons vu à la question a que l'ouverture d'un circuit virtuel s'effectue *de bout en bout* : seul l'ETTD appelé peut répondre à la demande d'ouverture de connexion envoyé par l'ETTD appelant. À la question d, nous voyons que la libération du circuit virtuel s'effectue *localement* : c'est le nœud auquel est raccordé l'ETTD qui confirme la libération de connexion et non l'ETTD distant. La libération d'un circuit virtuel s'effectue donc beaucoup plus rapidement que son ouverture.

e La détection des erreurs ne peut se faire au niveau de l'entité qui gère le circuit virtuel puisque le protocole de gestion du circuit virtuel ne contient aucun mécanisme de détection des erreurs. Elle est assurée au niveau du protocole gérant la liaison de données, grâce au FCS placé à la fin de la trame. Si la transmission altère une trame (et donc le paquet qui est dans son champ de données), celle-ci est rejetée car le récepteur détecte l'erreur dans la trame et demande sa retransmission. Tout paquet se trouvant dans une trame erronée est ignoré du récepteur.

EXERCICE 7 TRANSFERT DE DONNÉES SUR PLUSIEURS CIRCUITS VIRTUELS

Énoncé

Soit un réseau utilisant le protocole *X.25*, dans lequel un serveur *C* initialise les deux circuits virtuels qui le relient à *A* et *N*. Le premier, ouvert entre *C* et *A*, utilise une fenêtre de 1 alors que celui entre *N* et *C*, ouvert en second, utilise une fenêtre de 2. Le transfert des données se déroule comme suit : *A* et *N* envoient au même instant deux paquets vers le serveur *C*. Lorsqu'il a reçu tous les paquets des ETTD, *C* leur répond en envoyant à chacun deux paquets, *en entrelaçant les paquets des 2 circuits virtuels* (c'est-à-dire *C* envoie le premier paquet vers *A* puis le premier paquet vers *N* puis le second paquet vers *A* et enfin le second paquet vers *C*). On s'intéresse au transfert des paquets de données entre les stations *A*, *N* et le serveur *C*.

Protocole réseau : le transfert des données s'effectue sur des CVP (circuits virtuels permanents[4]), ouverts par le centre serveur *C*, au moment de la mise en service du réseau.

Les fenêtres de chaque CVP sont indiquées ultérieurement. La figure 3.22 montre deux stations clientes (*A* et *N*) communiquant avec un serveur *C*. Les points N_A, N_C, N_N constituent les points d'accès au réseau *X.25* (ce sont les ETCD locaux respectivement raccordés aux ETTD à *A*, *C* et *N*).

☞

4. Un CVP est un circuit virtuel établi une fois pour toutes, entre deux ETTD, jusqu'à la fin de l'abonnement contracté. On n'a pas besoin de l'établir, ni de le libérer.

Énoncé (suite)

Figure 3.22

Connexions entre ETTD et ETCD du réseau.

a Quels sont les numéros de voie logique utilisés par les différents ETTD, sachant que le plus grand numéro de voie logique disponible sur tous les ETTD est égal à 10 et qu'aucun ETTD n'a de circuit virtuel déjà ouvert ?

b Représentez, sur le même diagramme, les échanges de paquets correspondant au transfert de données entre les trois stations et répondant aux hypothèses décrites précédemment. Vous supposerez pour cela que les circuits virtuels sont déjà ouverts avant le transfert de données. Le circuit virtuel entre *A* et *C* utilise une fenêtre de 1, celui entre *N* et *C* une fenêtre de 2.

c Sur un circuit virtuel, les données peuvent aussi être acquittées de bout en bout, c'est-à-dire que l'ETTD destinataire est le seul habilité à acquitter les données reçues. Déterminez en quoi cela peut affecter la façon de transférer les données et les performances attendues.

Solution

a D'après les conventions d'affectation des numéros de voie logique, le circuit virtuel entre *A* et *C* utilise le numéro de voie logique 10 dans l'ETTD *C*. Le numéro 9 identifie le circuit virtuel entre *C* et *N*. Les ETTD *A* et *N* utilisent tous les deux, pour identifier le circuit virtuel avec *C*, le numéro de voie logique 1.

b La figure 3.23 décrit les échanges entre les trois ETTD.

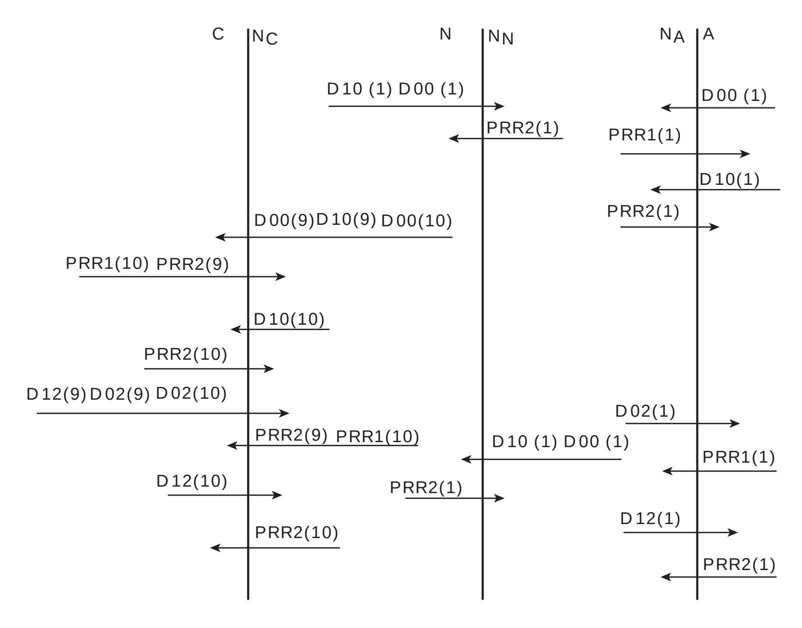


Figure 3.23

Transfert de données entre les trois ETTD.

c Avec des *acquittements locaux*, les paquets de données émis par l'ETTD sont acquittés par le nœud d'accès. Cette technique permet d'émettre les données le plus vite possible mais, en cas de panne, il y a un risque d'en perdre : dès qu'un paquet est acquitté, l'ETTD émetteur efface les données qui viennent d'être envoyées pour réutiliser la zone mémoire alors que le paquet n'est pas encore parvenu au destinataire ! (Comme pour l'envoi d'une

lettre par la Poste : ce n'est pas parce qu'on vient de la mettre dans la boîte qu'elle se trouve déjà dans les mains du destinataire. Elle peut se perdre ou être détruite avant d'arriver, même si cette éventualité est rare.)

Avec des *acquittements de bout en bout*, les nœuds intermédiaires se contentent de transporter le paquet puis de renvoyer l'acquittement qui est émis par le récepteur distant. En le recevant, l'émetteur est sûr que le paquet est bien parvenu à destination.

L'acquittement de bout en bout est la méthode d'acquittement la plus fiable mais aussi la plus lente. Après l'envoi du paquet dans le réseau, il faut attendre que l'acquittement du destinataire ait parcouru tout le réseau en sens inverse jusqu'à l'expéditeur. Au mieux, le transfert des données avec acquittement de bout en bout pour chaque paquet est deux fois plus lent qu'un transfert n'utilisant que des acquittements locaux. Si nous conservons notre exemple de courrier postal, cette méthode d'acquittement correspond à l'envoi d'une lettre recommandée avec accusé de réception.

Remarque

Pour éviter l'engorgement du réseau et des cadences de transfert trop lentes, il est recommandé de ne pas utiliser uniquement l'acquittement de bout en bout mais de le réserver à des points de reprise critiques du fichier. L'exercice 8 montre comment panacher acquittements locaux et acquittements de bout en bout.

Pour illustrer ce principe, supposons qu'on veuille transférer le contenu de ce livre. Sachant que chaque page génère plusieurs paquets de données, plusieurs manières de transférer le fichier peuvent s'envisager.

Chaque paquet est acquitté de bout en bout : cette méthode, longue et fastidieuse, n'est pas à conseiller, même si elle est très fiable !

Chaque page est acquittée de bout en bout : seul le dernier paquet de la page est acquitté de bout en bout. En recevant l'acquittement correspondant, l'émetteur sait que toute la page a été bien reçue, puisqu'un acquittement valide également les paquets qui le précèdent. Cette méthode est intéressante si le réseau présente une fiabilité insuffisante en regard des besoins de l'application de transfert de fichiers.

Chaque chapitre est acquitté de bout en bout : la méthode est la même pour le dernier paquet du chapitre au lieu du dernier paquet de la page. Cette méthode est certainement la plus sûre et la plus rapide si le réseau est suffisamment fiable

Exercice 8 Transfert de données sur des circuits virtuels avec acquittements locaux et acquittements de bout en bout

Énoncé

Un serveur *S* communique avec une station *A*. Il utilise pour cela un circuit virtuel *déjà ouvert*, identifié par les numéros de voie logique 7 du côté de *S*, 1 du côté de *A* ; la fenêtre du circuit virtuel vaut 2 dans les deux sens.

A envoie 4 paquets de données vers le serveur, qui répond par 2 paquets de données. L'acquittement des paquets de données est *local* pour les premiers paquets de données, *de bout en bout* pour le dernier paquet transmis par chaque ETTD.

a Représentez, à l'aide d'un diagramme, les échanges de paquets correspondant à la transaction effectuée entre les stations *A* et *S*. Les paquets transmis de bout en bout seront distingués des paquets acquittés localement par un astérisque sur le diagramme.

b Combien *A* peut-il encore envoyer de paquets, une fois qu'il a envoyé son dernier paquet de données ?

Solution

a La figure 3.24 montre le transfert des données entre *A* et *S*.

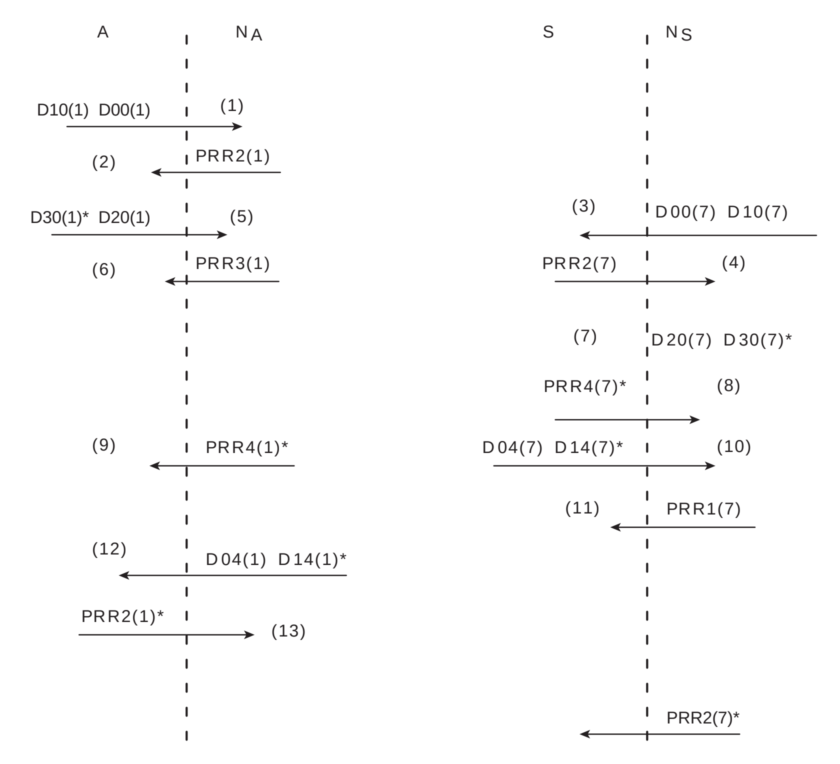


Figure 3.24

Transfert de données avec acquittements locaux et de bout en bout.

(1) : A envoie les paquets correspondant à la fenêtre d'anticipation.

(2) : L'ETCD les acquitte localement.

(3) : Après avoir traversé le réseau, les deux premiers paquets de données sont transmis à S.

(4) : S acquitte les paquets reçus.

(5) : A envoie ses 2 derniers paquets et demande l'acquittement de bout en bout du dernier paquet.

(6) : L'ETCD n'acquitte que le troisième paquet.

(7) : Les derniers paquets de A sont transmis à S.

(8) : S acquitte tous les paquets et demande la propagation de bout en bout de son acquittement.

(9) : L'acquittement des paquets a traversé tout le réseau et parvient à A.

(10) : S envoie 2 paquets et demande l'acquittement de bout en bout du dernier paquet, tout en validant les paquets reçus précédemment.

(11) : Seul le premier paquet de S peut être acquitté par l'ETCD.

(12) : Les paquets sont transmis à A ; le dernier doit être acquitté de bout en bout.

(13) : A acquitte les paquets et demande la propagation de bout en bout de son acquittement.

(14) : L'acquittement de bout en bout parvient à S.

b Après avoir envoyé son dernier paquet, *A* ne peut pas envoyer plus d'un paquet, car la réception de l'acquittement local *PRR3(1)* lui indique que seuls 3 paquets sont acquittés. La fenêtre n'est complètement ouverte qu'après réception de l'acquittement de bout en bout provenant de *S*. La situation est identique pour le serveur *S* qui n'a reçu que l'acquittement (local) du premier paquet.

Remarque

La demande d'acquittement de bout en bout revient au fait que l'ETTD distant commande l'ouverture de la fenêtre de l'ETTD local.

Chapitre 4

Les architectures de communication

Ce chapitre décrit comment se formalisent les notions d'empilement de couches de protocoles et de services, que nous avons rapidement évoquées au cours du chapitre précédent. Nous abordons également les modifications qu'il a fallu apporter au modèle initial pour décrire la structure des réseaux locaux (LAN), aujourd'hui les plus utilisés dans les entreprises ou même chez les particuliers. Nous introduisons ensuite l'architecture de communication utilisée dans Internet, connue sous le nom d'*architecture TCP/IP* ou encore de *pile TCP*. Enfin, nous évoquons comment normes et standards sont élaborés par les organismes internationaux et le comité technique chargé de l'évolution d'Internet.

1 Concept d'architecture en couches

L'empilement des couches et les services qu'elles offrent constituent l'*architecture de communication*. Une architecture de communication est donc une représentation abstraite (indépendante de toute référence à des logiciels ou des matériels particuliers) de la circulation des informations et des concepts utilisés au sein d'un réseau quelconque. Pour cela, nous nous appuyons sur le modèle abstrait d'architecture de communication défini dans le milieu des années 1970 pour les réseaux grande distance (WAN). Cette architecture, communément appelée *modèle de référence OSI* ou *modèle OSI*, a apporté un vocabulaire et des concepts encore utilisés de nos jours. En effet, même des architectures élaborées en dehors de ce modèle se réfèrent à la terminologie qui y est définie.

1.1 Pourquoi utiliser une architecture en couches ?

La structuration en couches considère un système comme logiquement composé d'un ensemble de *n* sous-systèmes ordonnés. Les sous-systèmes adjacents communiquent à travers leur interface commune. Un sous-système de rang *i* peut être constitué d'une ou plusieurs *entités* ; il communique avec les autres sous-systèmes de même rang : on parle alors de la *couche de rang i* ou, plus simplement, de la *couche i*. Les avantages d'une telle structure sont multiples :

- Une architecture de communication se définit entièrement en décrivant les services offerts par chaque couche, les interfaces entre les couches adjacentes et la manière dont ces couches coopèrent avec les entités du même niveau (les entités *homologues*) dans les autres systèmes.
- On peut développer séparément et simultanément toutes les couches d'une architecture de communication, une fois définies les interfaces entre les différents sous-systèmes.
- Le nombre d'interfaces à définir est minimal : il suffit de décrire, pour chaque niveau, les interfaces avec la couche supérieure (sauf pour la couche la plus élevée de l'architecture) et avec la couche inférieure (sauf pour la couche la plus basse). Les coopérations entre entités homologues sont régies par un ou plusieurs *protocoles*.

Ainsi, chaque couche fournit des services aux entités des couches supérieures et s'appuie sur les services offerts par les entités des couches inférieures. La couche la plus élevée offre à l'utilisateur tous les services utilisables dans l'architecture ; la couche la plus basse communique directement avec le support de transmission.

1.2 Terminologie employée

Dans la suite, nous utilisons la notation (*i*) pour signifier « de niveau *i* », afin de ne pas alourdir inutilement les définitions et les notations employées.

- *Service* (*i*). Capacité que possède la couche (*i*) et les couches inférieures à celle-ci, fournie aux entités (*i* + 1), à la frontière entre la couche (*i*) et la couche (*i* + 1). Les services sont invoqués par des *primitives*, spécifiques du service.
- *Primitive.* Demande de service par une entité de niveau supérieur à une entité de niveau inférieur.
- *Protocole* (*i*). Ensemble de règles et de formats déterminant les caractéristiques de communication des entités (*i*) lorsqu'elles effectuent les fonctions nécessaires à l'exécution du service (*i*). Le protocole utilise des unités de données appelées *PDU(i)* [*Protocol Data Unit(i)*].



- *Point d'accès à des services (i)* [*SAP(i)*, *Service Access Point(i)*]. Point où les services (*i*) sont fournis par une entité (*i*) à une entité (*i* + 1). Les unités de données du service *SDU*(*i* + 1) [SDU, *Service Data Unit*] traversent les *SAP(i)*.
- *SDU*(*i* + 1). Unités de données du service échangées localement entre entités (*i* + 1) et entités (*i*) pour l'exécution d'un service (*i*).
- *PDU(i)*. Unité de données du protocole, échangée entre entités (*i*) homologues.

1.3 Notion d'encapsulation

Une entité (*i* + 1) qui fait appel aux services assurés par une entité (*i*) lui transmet les données concernées et utilise la primitive appropriée à la demande d'exécution du service. Pour cela, l'entité (*i*) construit une (ou plusieurs) unité(s) de données, pour transporter les données, conformément au protocole (*i*). Celles-ci sont insérées dans le champ de données de la PDU(*i*) : on dit que les données (*i* + 1) sont *encapsulées* dans une unité de données (*i*).

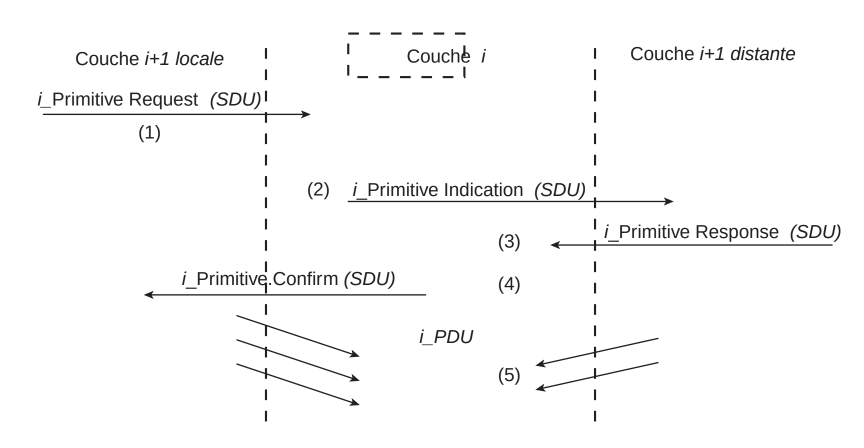
Une entité (*i*) reçoit donc les données émanant de l'entité (*i* + 1) sous la forme d'une SDU(*i*). À chaque SDU(*i*), l'entité (*i*) ajoute – en tête et/ou en queue – des informations de contrôle permettant d'exécuter le service demandé pour constituer une PDU(*i*) qui est soumise à la couche inférieure. Ainsi, une PDU(*i*) constitue une SDU(*i* – 1) à l'interface entre les couches (*i*) et (*i* – 1). Inversement, une SDU(*i*) contient une PDU(*i* + 1).

Dans certaines couches, les protocoles employés pour rendre le service (*i*) peuvent utiliser plusieurs PDU(*i*) pour traiter une SDU(*i*). En outre, les services peuvent nécessiter des PDU spécifiques comme, par exemple, des PDU d'acquittement qui ne contiennent pas de champ de données.

Une couche fournit un ensemble de services au niveau supérieur, invoqués par des primitives. On utilise quatre primitives, selon le sens et la nature de l'interaction : la requête, l'indication, la réponse et la confirmation (voir figure 4.1).

Figure 4.1

Les différents types de données échangées entre couches adjacentes ou entre couches homologues.



Comme on le voit à la figure 4.1, la partie données de la primitive est la SDU : son contenu est totalement transparent pour le fournisseur du service. Pour signaler un événement à la

couche supérieure lorsque le protocole le prévoit, le fournisseur utilise une primitive de type *Indication*. Le destinataire de la primitive renvoie une primitive de type *Response* (si le protocole le requiert). Enfin, une primitive de type *Confirm*, émise par son prestataire de service, permet à l'auteur de la requête d'être informé de la fin – correcte ou non – de sa requête. Un paramètre précise alors les conditions de l'exécution de la requête.

2 Modèle OSI (*Open System Interconnection*)

La compatibilité (l'interopérabilité) entre équipements hétérogènes (constructeurs, fonctions ou générations de matériels différents…) implique des normes d'interconnexion définissant le comportement de chaque équipement vis-à-vis des autres. Tout équipement (ou ensemble d'équipements) à interconnecter est un *système ouvert* (un ordinateur, un terminal, un réseau…), s'il respecte des normes d'interconnexion. Le modèle OSI est une architecture abstraite de communication, décrit dans la norme *X.200* de l'ITU. Il est composé de sept couches, chacune remplissant une partie bien définie des fonctions permettant l'interconnexion.

Remarque

Open System Interconnection se traduit en français par « interconnexion des systèmes ouverts », ce qui donne l'acronyme ISO. Pour éviter toute confusion entre le modèle et l'organisme de normalisation, nous parlerons du modèle OSI.

2.1 Différentes couches du modèle OSI

La figure 4.2 montre les SDU et PDU de toutes les couches du modèle OSI. La couche de plus bas niveau est la couche *Physique*. Elle se caractérise par son taux d'erreurs, la vitesse de transmission et le délai de transit. L'unité de données manipulée à ce niveau est le *bit*. Au-dessus, la couche *Liaison de données* fournit les moyens d'établir, de maintenir et de gérer les connexions de liaison de données entre entités de réseau. Elle détecte et corrige, dans la mesure du possible, les erreurs de la couche physique. La *trame* est l'unité de données manipulée par la liaison de données. La couche *Réseau* fournit aux entités de transport les moyens d'établir, de maintenir et de gérer les connexions de réseau ; elle manipule des *paquets* et les achemine à travers le réseau. Ces trois premières couches ont été définies dans la norme *X.25*, que nous avons évoquée au chapitre 3. Tous les équipements du réseau, dans les systèmes intermédiaires comme dans les systèmes d'extrémité, contiennent des entités de réseau. Seuls les systèmes d'extrémité implémentent les couches supérieures. La quatrième couche, *Transport*, assure un transfert de données fiable et optimise les coûts d'utilisation des services réseau disponibles, compte tenu des exigences de service des entités supérieures. Le *message* est l'unité de données qu'elle manipule. Cette couche charnière masque, pour les couches hautes, les disparités entre réseaux. La cinquième couche, *Session*, organise et synchronise le dialogue entre les systèmes d'extrémité. La sixième couche, *Présentation*, s'occupe de la représentation des informations, quels que soient les modes de représentation interne des machines et dans le réseau. Elle peut se charger aussi de la compression de données et de la sécurité des informations échangées (chiffrement/déchiffrement, qu'on appelle parfois cryptage/décryptage). La dernière couche est la couche *Application*. Elle contient les entités d'application, c'est-à-dire les processus des utilisateurs

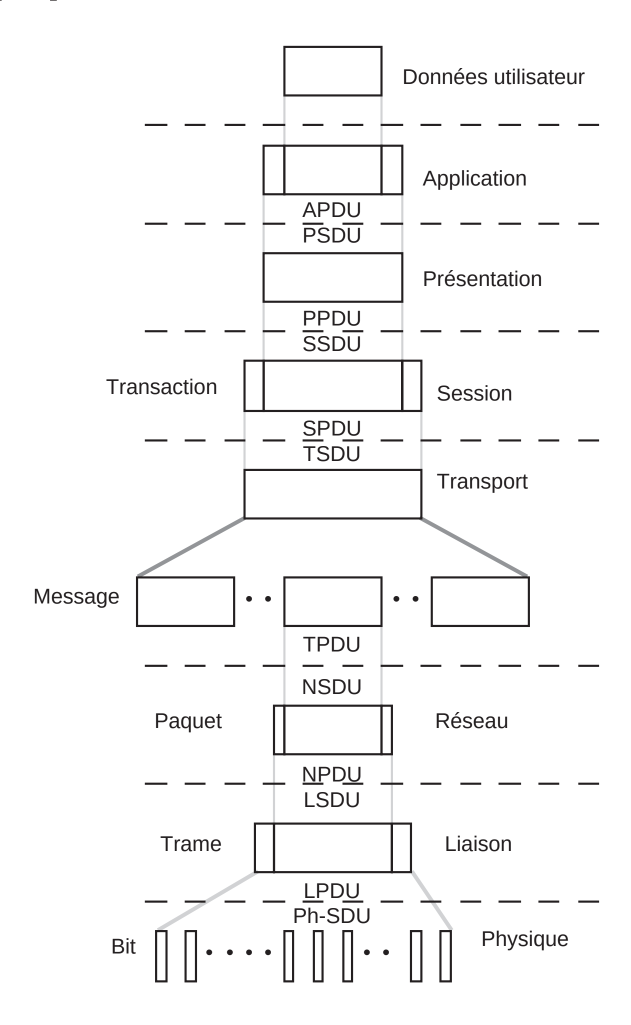


qui génèrent les informations à échanger. Au sens du modèle OSI, une entité d'application peut être une entité de messagerie ou de transfert de fichiers par exemple. Il n'y a pas de mot dans la langue française pour qualifier l'unité de données des trois dernières couches. On parle en général de *messages*.

Dans les échanges à l'intérieur du modèle OSI, nous trouvons des PDU et des SDU : *Ph_PDU* désigne les PDU du niveau Physique, *L_PDU* celle du niveau Liaison, les *N_PDU* (*N* pour network) sont les *PDU* de niveau Réseau… La lettre préfixe est *T* pour Transport, *S* pour Session, *P* pour Présentation et *A* pour Application. De même, nous rencontrons des SDU de niveau Liaison, de niveau Réseau, etc. Les premières sont des *L_SDU* et les suivantes des *N_SDU*, etc. À titre d'exemple, les *N_SDU* sont des unités de données provenant d'entités de la couche Transport, comme les paramètres de connexion fournis par l'entité de transport pour l'ouverture du circuit virtuel au niveau Réseau.

Figure 4.2

Empilement des sept couches du modèle OSI, avec les PDU et SDU utilisées.



A priori, chaque couche (sauf la couche Physique) peut utiliser les quatre types de primitives. Dans le modèle de référence, le nom complet d'une primitive est couramment obtenu en faisant précéder son type par l'initiale de la couche, suivie de la nature de l'opération : *Request, Indication, Response, Confirm*.

Exemple

La primitive *T_CONNECT.Request* est la requête (*Request*) de demande d'ouverture de connexion (*CONNECT*) émise par l'entité de couche Session vers son entité de Transport (*T*). Cette primitive demande à la couche Transport d'établir une connexion.

2.2 Modèle d'architecture à 5 couches

Le découpage proposé par le modèle initial a connu une évolution, du fait de la complexité des normes. En effet, les couches 5 et 6 (Session et Présentation) ont été progressivement vidées de leur substance au fil du temps : les couches adjacentes ont intégré leurs fonctionnalités. La couche Transport assume le plus souvent celles de la couche Session, alors que les applications incorporent la description des structures de données, même si la compatibilité totale entre systèmes ou versions de systèmes est encore loin d'être parfaite… Les architectures existantes ne respectent plus vraiment le découpage tel que le décrit le modèle initial, mais le principe même de la structuration en couches reste la base de toutes les architectures. La terminologie et le découpage des services réseau ont été adoptés par tous : on parle couramment de couche 2 pour la couche Liaison, de couche 3 pour désigner la couche Réseau, même si l'architecture considérée ne les place à cet endroit !

Remarque

Dans l'architecture ATM (que nous avons citée au chapitre 3), il n'y a que deux couches basses : la couche 1 ou Physique et… la couche 3 ou couche ATM. La couche supérieure, baptisée *Adaptation à l'ATM* (ou AAL, *ATM Adaption Layer*) est considérée comme une couche Transport donc de niveau 4.

2.3 Modèle d'architecture adapté aux réseaux locaux

L'IEEE (*Institute for Electricity and Electronics Engineers*), une société américaine, a constitué un comité d'études au début des années 1980 (le groupe 802, essentiellement des constructeurs américains). L'objectif est alors de développer des standards pour la transmission de messages à haut débit entre systèmes informatiques, à travers un support partagé par ces systèmes et indépendant de leur architecture. Ce comité a publié une série de standards nommés 802.*n*. Nous les décrivons plus en détail au chapitre 5. L'ISO a ensuite repris les travaux du groupe 802 et les a référencés sous le numéro 8802.*n* (le *n* des références ISO est identique au *n* utilisé dans les références de l'IEEE).

La modélisation de l'IEEE redéfinit les niveaux 1 et 2 du modèle OSI pour les réseaux locaux. Cette modélisation spécifie les services rendus à la couche supérieure et la façon d'implanter les niveaux 1 et 2. La figure 4.3 montre la correspondance entre les couches 1 et 2 du modèle OSI et les couches du modèle IEEE. Nous remarquons que, par rapport au modèle OSI, l'architecture normalisée dans les réseaux locaux découpe la couche Liaison en deux sous-couches : MAC (*Medium Access Control*) et LLC (*Logical Link Control*).

Le niveau MAC, comme son nom l'indique, définit la *méthode d'accès*, c'est-à-dire la manière dont il faut envoyer et recevoir les données sur le support partagé par toutes les stations du réseau local. Il existe différentes méthodes d'accès, incompatibles entre elles. CSMA/CD, la méthode d'accès des réseaux Ethernet, est la plus connue et la plus utilisée. Elle est décrite dans le standard 802.3. La sous-couche LLC masque les disparités des méthodes d'accès. Le chapitre 5 consacré aux réseaux locaux décrit le fonctionnement détaillé de ces deux sous-couches.



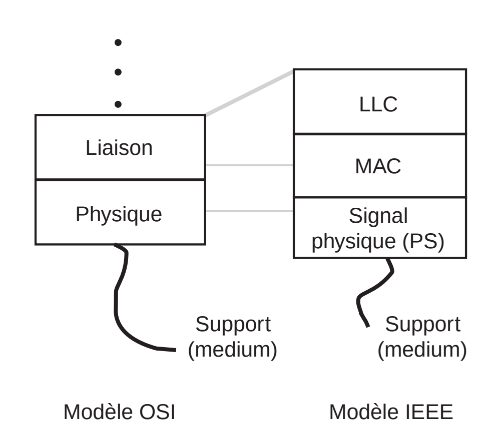


Figure 4.3

Comparaison des modèles OSI et IEEE : des positionnements différents.

3 Architecture TCP/IP (*Transmission Control Protocol/Internet Protocol*)

L'architecture TCP/IP porte le nom des protocoles principaux qui la constituent, à savoir TCP et IP ; on l'a définie dans les années 1960 pour le réseau ARPAnet. Elle s'est considérablement développée avec le succès d'Internet. Les chapitres 6 et 7 consacrés à Internet et à chacun de ses deux protocoles phares TCP et IP en détaillent le fonctionnement.

3.1 Standardisation de l'architecture TCP/IP

La conception de l'architecture TCP/IP a été très différente de la méthode utilisée dans le modèle OSI. *L*es organismes de normalisation internationaux ont en effet défini des concepts universels, répondant à tous les besoins possibles, et des fonctionnalités en dehors de tout souci de réalisation. Les normes de l'ISO sont de ce fait très complexes, car elles contiennent de nombreuses options, destinées à couvrir l'ensemble des fonctionnalités proposées, quel que soit l'environnement d'application. Les concepteurs de l'architecture TCP/IP se sont attachés à décrire en premier les protocoles, avant de proposer un modèle de référence décrivant leur empilement. De plus, ils ont privilégié une approche pragmatique : trouver une solution rapide et opérationnelle, même si elle ne résout pas la totalité du problème.

3.2 Présentation de l'architecture TCP/IP

L'architecture TCP/IP compte quatre couches, comme le montre la figure 4.4 : *Réseau physique*, *Réseau*, *Transport* et *Application.*

Aucune caractéristique particulière n'est requise pour l'infrastructure du ou des réseaux physiques traversés. La couche Réseau physique est donc quelconque. La couche Réseau (qu'il serait d'ailleurs plus juste de dénommer « interréseaux ») assure la communication entre les réseaux grâce au protocole IP (*Internet*[1] *Protocol*). On utilise la commutation de

1. Internet est l'apocope d'*internetworking* (interréseaux).

paquets de type datagramme pour acheminer des données entre les systèmes d'extrémité, quelle que soit la technologie réseau qu'ils emploient. Le protocole IP gère les datagrammes : il les achemine jusqu'à leur destinataire, se charge du routage et de l'adaptation de la taille des données au réseau sous-jacent. IP définit en fait un service minimal, l'acheminement des datagrammes à travers l'interconnexion de réseaux. Ce service est sans connexion et sans garantie. Il ne fait aucune hypothèse quant à la fiabilité des réseaux traversés.

Au-dessus de la couche IP, l'architecture définit deux protocoles de transport, en fonction des besoins des utilisateurs : un protocole en mode connecté, TCP (*Transmission Control Protocol*), et un protocole en mode non connecté, UDP (*User Datagram Protocol*). TCP est destiné à fiabiliser les échanges : il permet aux utilisateurs situés aux extrémités de la connexion d'échanger des données, avec contrôle de flux, contrôle d'erreur, contrôle de séquence entre les deux extrémités. Il garantit en particulier la livraison séquentielle des données, leur non-duplication et la récupération des données manquantes. UDP est un protocole non fiable. Il sert aux applications qui ne souhaitent pas ralentir les transferts de données par la lourdeur de la mise en œuvre des processus de gestion du mode connecté, ou à celles qui n'ont pas besoin de la fiabilité de TCP : inutile de mettre en œuvre des mécanismes complexes pour garantir le séquencement des messages par exemple, quand il n'y a qu'un seul message à émettre.

Enfin, la couche Application se greffe directement au-dessus de la couche Transport. Elle contient tous les protocoles de haut niveau qu'un utilisateur souhaite avoir à sa disposition : Telnet, FTP, SMTP, HTTP… Nous verrons plusieurs exemples d'applications au chapitre 9.

Figure 4.4

Architecture TCP/IP : un ensemble charnière TCP et IP.

Applications
Transport (*UDP* et *TCP*)
Interréseaux (*IP*)
Réseau physique

4 Normalisation dans les télécommunications et les réseaux

Dans des domaines techniques comme les réseaux et les télécommunications, la normalisation répond aussi bien aux attentes des consommateurs qu'aux besoins des fabricants. D'un côté, elle offre aux utilisateurs la garantie que deux produits aux fonctions identiques mais de fabricants différents puissent fonctionner correctement ensemble. De l'autre, les industriels peuvent espérer toucher un plus grand nombre de consommateurs grâce à la normalisation de leurs produits. En effet, toute solution propriétaire provoque la réticence des utilisateurs et des entreprises à dépendre d'un seul fournisseur pour leur approvisionnement : dans la mesure où certains équipements sont vitaux pour la survie même de l'entreprise, la continuité du service passe par la possibilité de disposer de plusieurs sources indépendantes d'approvisionnement.



Différents organismes de normalisation édictent des avis qui couvrent tous les aspects d'un équipement : aspects électriques, mécaniques, interconnexion… Les principaux organismes internationaux de normalisation regroupent des représentants des industriels, des administrations, des utilisateurs : l'ISO (*International Standardization Organization*), l'ITU (*International Telecommunications Union*)… On trouve également divers groupements de constructeurs comme : l'ECMA (*European Computer Manufacturer*), l'EIA (*Electronic Industries Association*)… Dans Internet, l'IAB (*Internet Architecture Board*) définit la politique du réseau à long terme alors que l'IETF (*Internet Engineering Task Force*) s'occupe de l'homogénéité des solutions et publie les RFC (*Request For Comments*).

Une norme passe par plusieurs étapes : le résultat des compromis entre les différentes parties s'exprime dans un document brouillon (*draft*). La forme stable du document est publiée sous forme de *draft proposable*. Le *Draft International Standard* est la forme quasiment définitive. Il constitue une base de travail pour les industriels. Enfin, l'IS (*International Standard*) est la forme définitive du document. L'organisme considéré publie la forme finale, en utilisant une référence qui dépend de son domaine d'application.

Résumé

Une architecture de communication formalise l'empilement de couches de protocoles et des services offerts pour assurer l'interconnexion des systèmes. Nous avons présenté le modèle abstrait, défini dans le milieu des années 1970, communément appelé *modèle de référence* ou *modèle OSI*. Si ce modèle ne précise pas le fonctionnement détaillé des systèmes réels, il a apporté un vocabulaire et des concepts de structuration encore utilisés de nos jours. La preuve en est donnée par les adaptations apportées au modèle OSI initial pour tenir compte de l'avènement des réseaux locaux, aujourd'hui les plus utilisés dans les entreprises ou chez les particuliers. Nous avons présenté succinctement l'architecture de communication TCP/IP, la plus largement répandue puisqu'elle est utilisée dans Internet. Les prochains chapitres détailleront les protocoles un par un. Enfin, nous évoquons brièvement le processus de normalisation.

Problèmes et exercices

EXERCICE 1 CHOIX DES PRIMITIVES D'UN NIVEAU DONNÉ

Énoncé Une interface entre deux couches adjacentes peut se décrire par l'ensemble des primitives qu'elle offre aux entités de la couche supérieure. Quels sont les critères à retenir dans le choix des primitives d'un niveau donné ?

Solution Trois considérations principales doivent guider le concepteur d'un niveau donné :

- définir les services offerts par la couche (i) aux entités ($i + 1$), en tenant compte de la position de ce niveau dans l'architecture de communication ;
- proposer un nombre minimal de primitives différentes, pour ne pas compliquer inutilement l'interface ;
- minimiser le nombre de paramètres à prendre en compte dans chaque primitive définie.

Pour satisfaire la première condition, il faut avoir une idée très précise du fonctionnement de la couche (i). Par ailleurs, la complexité de l'interface dépend de l'endroit où se situe le niveau considéré : une couche de bas niveau offre forcément des services plus limités qu'une couche haute de l'architecture. Il faut donc définir les services qui seront disponibles et décider du nombre d'entités nécessaires à leur gestion. Il faut également déterminer les interactions entre les différentes entités, afin de définir la circulation des informations au sein de la couche.

La deuxième condition doit être remplie dans un souci d'efficacité. Si l'interface compte un grand nombre de primitives différentes, l'entité ($i + 1$) risque au mieux de ne pas en utiliser toutes les subtilités. Au pire, elle peut entraîner une baisse des performances préjudiciable à toute l'architecture. En effet, le nombre élevé de primitives risque fort de mener à des doublons. À l'opposé, les entités ($i + 1$) pourraient laisser certaines primitives inusitées.

La troisième condition fournit une meilleure lisibilité du service demandé et accélère son traitement. Supposons que chaque paramètre de la primitive soit géré par une entité. Si la primitive contient 10 paramètres, il faudra concevoir 10 entités pour les manipuler, définir leur mode de coopération et prévoir toutes les situations possibles entre chaque paire d'entités concernées par le traitement du service… Cela risque de provoquer, à l'intérieur du niveau, des boucles dans le parcours de données entre les entités chargées de traiter le service demandé. Dans tous les cas, le temps de traitement de la primitive s'en trouve notablement allongé par rapport à une primitive plus simple. Il faut donc assurer un compromis entre des conditions contradictoires : une primitive simple se traite plus efficacement, mais elle n'exécute qu'un service limité par rapport à une primitive plus sophistiquée. Il faut alors augmenter le nombre de primitives disponibles. On risque de tomber dans le travers de définir un excès de primitives, forcément très mal utilisé…

Exercice 2 Procédures en cas de panne du réseau

Énoncé

Pendant un transfert de données sur un circuit virtuel, une panne se produit à l'intérieur d'un réseau respectant la norme *X.25*.

a Quelles sont les conséquences de la panne dans le réseau ?

b Quels types de primitives le réseau utilise-t-il pour signaler l'événement aux deux extrémités du circuit virtuel ?

c Quelles en sont les conséquences pour les entités de niveau Transport ?

d Quelles sont les stratégies de reprise possibles ?

Solution

a La panne dans le réseau provoque la rupture du circuit virtuel déjà établi. Les nœuds voisins du nœud défaillant (ou les nœuds situés aux deux extrémités de la liaison de données défaillante) détectent cet événement et le propagent jusqu'aux extrémités du circuit virtuel.

b Les entités de Réseau des systèmes d'extrémité le signalent à leurs entités de Transport en envoyant une primitive de type Indication, indiquant la cause de la rupture du circuit virtuel, lorsque celle-ci est connue.

c En recevant la primitive de type Indication, les entités de Transport réagissent en fonction des choix qu'elles ont faits au moment de l'établissement de la connexion de Transport. Par exemple, si elles ont ou non négocié la possibilité d'un fonctionnement en mode dégradé.

d Deux stratégies sont possibles : soit il y a abandon de la communication interrompue, soit il y a tentative de reprise, en particulier si la connexion de Transport a prévu un fonctionnement en mode dégradé.

Dans le premier cas, la connexion est rejetée ; il faut rétablir ultérieurement une connexion de Transport, en utilisant la même connexion Réseau (si on utilise un CVP, circuit virtuel permanent, réinitialisé) ou en ouvrant une nouvelle connexion Réseau avant d'établir la nouvelle connexion de Transport.

Dans le second cas, les entités de Transport ont prévu d'utiliser une procédure de récupération d'erreurs, ce qui leur permet de redémarrer le transfert des données par la réémission de toutes les données non encore acquittées.

Exercice 3 Établissement de connexions Liaison et Réseau

Énoncé

Une entité de Réseau souhaite envoyer des données provenant des couches supérieures.

a Décrivez les échanges de primitives entre une entité de Réseau quelconque et l'entité de Liaison, en supposant que la liaison de données soit exploitée avec le protocole LAP-B, qu'elle ne soit pas encore initialisée et qu'il n'y ait aucune erreur de transmission des données.

b Que se passe-t-il s'il se produit une erreur de transmission ?

c En supposant que la liaison de données soit opérationnelle, décrivez les échanges entre l'entité de Réseau locale et l'entité de Liaison locale pour ouvrir un circuit virtuel.

Solution **a** Supposons que nous soyons du côté du système d'extrémité qui prend l'initiative. Nous appellerons *entité de Réseau locale* l'entité de Réseau située dans notre système et *entité de Réseau distante* l'entité de Réseau du système d'extrémité qui doit recevoir les données. De même, nous appellerons *nœud local* le point d'accès au réseau du système d'extrémité local et *nœud distant* le point d'accès au réseau du système d'extrémité distant. Les échanges entre les entités locales de Liaison et de Réseau concernent le système d'extrémité local qui désire envoyer des données.

Pour envoyer ses données à l'entité de Réseau distante, l'entité de Réseau locale émet une primitive de type Requête vers son entité locale de Liaison. Cette requête contient les données à transmettre. À la réception de cette primitive, l'entité de Liaison locale constate qu'elle ne peut pas assurer le service demandé puisque la liaison de données n'est pas opérationnelle. Elle établit donc la liaison à l'aide d'une trame d'initialisation (une trame U, par exemple SABM,P et reçoit en retour la trame U d'acceptation UA,F, [voir chapitre 2]). Puis l'entité de Liaison locale envoie à l'entité de Liaison du nœud local autant de trames I que de paquets à envoyer. Elle reçoit les acquittements des trames I émises par l'entité de Liaison du nœud local, conformément au protocole de liaison. Une fois le transfert de données correctement terminé, l'entité de Liaison locale envoie à l'entité de Réseau locale une primitive de type Indication pour la prévenir de la (bonne) fin du transfert. La figure 4.5 montre les échanges entre les deux entités locales et l'entité de Liaison du nœud local.

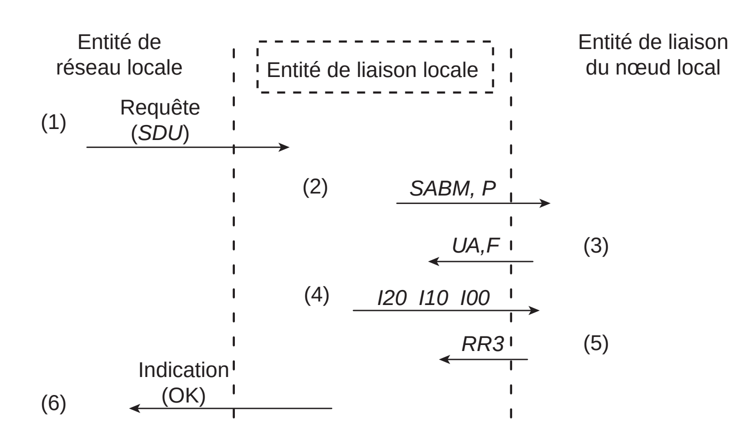


Figure 4.5

Échanges entre les différentes entités pour le traitement de la requête : une requête réseau simple peut donner lieu à de multiples échanges du protocole de liaison.

(1) L'entité de réseau locale du système d'extrémité émet une requête pour envoyer les données (SDU).

(2) L'entité de liaison locale demande l'initialisation de la liaison de données en envoyant la trame SABM,P

(3) l'entité de liaison du nœud d'accès accepte l'initialisation de la liaison en envoyant UA,F.

(4) L'entité de liaison d'accés émet autant de trames I que nécessaire (ici 3 trames).

(5) L'entité de liaison du nœud local acquitte les trames I.

(6) L'entité de liaison locale prévient l'entité de réseau locale de la bonne fin du transfert de données.

b En cas d'erreur de transmission, l'entité de Liaison située dans le nœud d'accès émet une trame REJ si la trame erronée est suivie d'une autre trame I. Si la trame erronée est la dernière, elle est retransmise par l'entité de Liaison locale après expiration du temporisateur associé.



Remarque

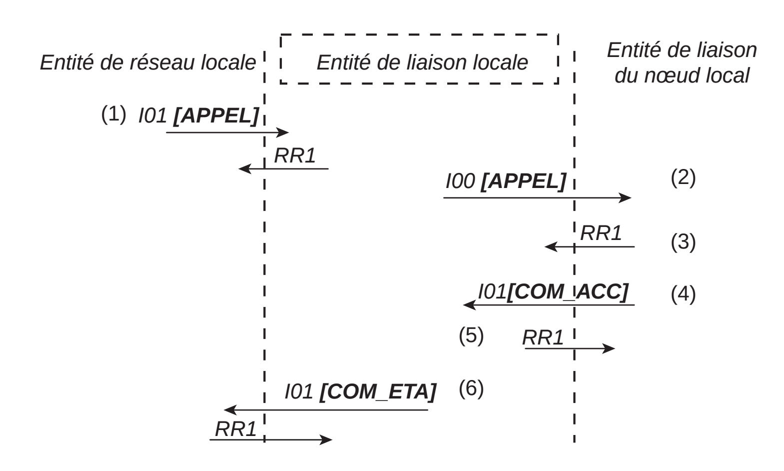
L'échange des trames et des primitives montre bien que les entités de Réseau (locale et distante) ne peuvent pas être averties d'une erreur de transmission, puisque cet événement est totalement pris en charge par leurs entités de Liaison respectives : les entités de Réseau ne manipulent que des unités de données exemptes d'erreurs (en cas d'erreurs résiduelles, elles gèrent des unités de données dont les erreurs n'ont pas été détectées par les entités de Liaison).

C Pour ouvrir le circuit virtuel, l'entité de Réseau locale envoie une primitive de type Requête avec les paramètres de la future connexion contenus dans le paquet *APPEL* (voir chapitre 3). L'entité de Liaison locale insère ce paquet dans une trame I et l'envoie dans le réseau. L'entité de Liaison du nœud local acquitte la trame I contenant le paquet *APPEL*.

Au bout d'un certain temps, l'entité de Liaison locale reçoit de l'entité de Liaison du nœud local la trame I contenant le paquet *COMMUNICATION_ETABLIE* (provenant de l'entité de Réseau distante). Elle extrait ce paquet du champ de données et le soumet à l'entité de Réseau locale. Celle-ci analyse le contenu du paquet et apprend ainsi que l'entité de Réseau distante a accepté la connexion. Elle peut alors commencer le transfert de données. La figure 4.6 décrit les échanges entre les entités concernées. Dans cette figure, les unités de données manipulées par l'entité de Réseau sont en italique gras ; le contenu du champ de données d'une trame est entre crochets. Les trames sont les unités de données manipulées par les entités de Liaison.

Figure 4.6

Échange de données entre les entités de Réseau et de Liaison.



(1) L'entité de réseau locale émet une demande d'ouverture de connexion (*APPEL*).
(2) L'entité de liaison locale insère le paquet *APPEL* dans une trame *I* référencée *I00.*
(3) L'entité de liaison du nœud local acquitte la trame contenant le paquet *APPEL.*
(4) L'entité de liaison du nœud local a reçu la réponse de l'entité de réseau distante. Elle l'insère dans le champ de données de la trame *I01.*
(5) L'entité de liaison locale acquitte la trame contenant la réponse de l'entité de réseau distante.
(6) L'entité de liaison locale soumet le résultat de la requête à l'entité de réseau locale par une primitive de type *Indication* (*COM_ETA*).

EXERCICE 4 NOTIONS DE SDU ET PDU DES NIVEAUX RÉSEAU ET LIAISON

Énoncé

Utilisez les notations de ce chapitre pour décrire les PDU et SDU échangées entre entités locales de couches adjacentes : reprenez l'exercice précédent demandant les échanges de données nécessaires à l'ouverture du circuit virtuel en indiquant quelles sont les PDU et les SDU des niveaux Réseau et Liaison de données.

Solution

Dans ces échanges, nous trouvons deux types de PDU et deux types de SDU *:* les PDU de niveau Liaison (*L_PDU*) et de niveau Réseau (*N_PDU*). Puisque nous nous plaçons au niveau de l'interface Liaison et Réseau, nous ne pouvons pas trouver de *N_SDU* dans notre échange.

Les *N_PDU* sont les unités de données manipulées par les entités de Réseau : il s'agit du paquet *APPEL* (généré par l'entité de Réseau locale à partir des données fournies par l'entité de Transport locale) et du paquet *COMMUNICATION_ETABLIE* (généré par l'entité de Réseau distante et transporté à travers le réseau de communication jusqu'à l'entité de Réseau locale).

Les *L_SDU* sont les unités de données du service fourni par la couche Liaison. Ces unités de données sont insérées dans le champ de données d'une trame d'informations. Elles contiennent les paquets *APPEL* et *COMMUNICATION_ETABLIE*.

Les *L_PDU* sont les unités de données que gère le protocole de liaison. Il s'agit donc des trames de type I, S et U. Dans notre exemple, nous trouvons les trames I transportant les paquets et les trames S qui acquittent les trames précédentes : d'une part les trames I00 et I01 (*L_PDU* avec champ de données) et, d'autre part, les trames RR1 (*L_PDU* d'acquittement, sans données).

EXERCICE 5 CONTENU DES PDU D'UN NIVEAU QUELCONQUE

Énoncé

a Que peut contenir une PDU(*N*), c'est-à-dire une PDU de niveau *N*, pour $N \leq 7$?

b Donnez des exemples pour $N = 2$ et $N = 3$ dans le cas de *X.25* (gestion des circuits virtuels avec LAP-B comme protocole de liaison).

Solution

a En nous appuyant sur l'exercice précédent, nous voyons qu'il y a deux types de PDU(*N*) : celles qui contiennent des informations de supervision (ou de commande) et celles qui contiennent des données et/ou des informations de commande de l'utilisateur. Les entités homologues s'échangent le premier type de PDU pour la supervision du protocole de niveau *N*. Le second type de PDU correspond à celles qui transportent des données et des informations de commande du niveau $N + 1$: ce sont les PDU qui transportent tout ou partie d'une SDU de niveau *N*, c'est-à-dire une PDU de niveau $N + 1$.

b Lorsque $N = 3$, il s'agit de *N_PDU.* Ce sont soit des PDU de commande, comme *APPEL*, *RR*, *LIB_CONF*, *COMMUNICATION_ETABLIE*, soit des paquets contenant les données de l'utilisateur ou des informations de commande provenant de la couche Transport.



Quand $N = 2$, les *L_PDU* de commande sont, par exemple, *SABM*, *UA*, *RR*… Les trames I sont les seules *L_PDU* contenant des données d'un utilisateur de la couche Réseau, quel que soit leur contenu.

Remarque

Puisque les trames I sont les seules *L_PDU* contenant des unités de données du protocole réseau, nous voyons bien maintenant la différence entre une *trame RR* et un *paquet RR :* une trame RR est une trame S, c'est-à-dire une *L_PDU* de commande. Un paquet RR est une *N_PDU* de commande transportée dans une trame I.

EXERCICE 6 QUELQUES PRIMITIVES DU MODÈLE OSI

Énoncé

Un système d'extrémité utilise une architecture de communication conforme au modèle OSI. Considérons par exemple la primitive : *N_CONNECT.Indication*.

a Dans quelle couche du modèle OSI se situe l'entité qui émet cette primitive ?

b À quelle entité est destinée cette primitive ?

c Considérons maintenant la primitive : *L_DATA.Request*. À quel niveau se situe l'entité qui émet cette primitive ? Quelle est son utilité ?

Solution

a La primitive provient d'une entité de Réseau. Elle signale qu'une demande d'ouverture de circuit virtuel lui est parvenue.

b La primitive s'adresse à une entité de Transport, qui décide si elle accepte ou non la connexion. La réponse de cette entité se trouve dans la primitive *N_CONNECT.Response* (elle contient un paramètre codant l'acceptation ou le refus de la connexion).

c La primitive *L_DATA.Request* émane d'une entité de Réseau, destinée à l'entité de Liaison. Elle fournit à cette dernière les paquets de données à émettre. En la recevant, l'entité de Liaison construit une trame d'informations avec les informations de commande appropriées (en-tête et queue de la trame), puis elle place le paquet dans le champ de données de la trame.

EXERCICE 7 RÔLE DES PRIMITIVES DU MODÈLE OSI

Énoncé

Un système d'extrémité utilise une architecture de communication conforme au modèle OSI. Considérons la primitive de service : *N_CONNECT.Request*.

a Dans quelle couche du modèle OSI se situe l'entité émettrice de la primitive ?

b À qui est destinée cette primitive ?

c À quoi sert cette primitive, si le système d'extrémité accède à un réseau public de données utilisant le protocole *X.25* ?

d Comment l'entité émettrice de la primitive *N_CONNECT.Request* saura-t-elle que sa primitive a été traitée ?

Solution

a L'entité émettrice se situe *au-dessus* de la couche Réseau puisqu'elle utilise ses services. Elle se trouve donc dans la couche Transport.

b Elle s'adresse au prestataire de services, donc à une entité de niveau Réseau.

c Elle sert à déclencher le processus d'établissement d'un circuit virtuel.

d Elle sait que sa primitive est prise en compte en recevant une primitive *N_CONNECT.Confirm* de la part de l'entité de niveau Réseau. Cette primitive lui indique comment s'est déroulé l'établissement de la connexion (succès ou échec).

EXERCICE 8 RELATIONS ENTRE PDU ET SDU DES DIFFÉRENTS NIVEAUX

Énoncé

Un utilisateur souhaite envoyer une photo de 448 000 octets dans un réseau. L'architecture de communication respecte le modèle à 5 couches évoqué à la section 2.2, c'est-à-dire un modèle de référence OSI adapté aux réseaux locaux. Sachant que la couche Transport gère des messages de 1 Kilo-octet (1 Kilo-octet vaut 1 024 octets) et utilise pour cela un en-tête de 24 octets pour gérer 1 000 octets de données, que la couche Réseau utilise des paquets de 128 octets dont les 3 premiers constituent l'en-tête du paquet, on demande :

a Quelle est, en octets ou en kilo-octets, la taille d'une *T_SDU ?* Quelle est celle d'une *T_PDU* de données ? D'une *N_SDU* ? D'une *N_PDU* de données ? D'une *L_SDU* ? D'une *L_PDU* de données ?

b Combien de paquets de données faut-il envoyer pour transmettre la photo si on ne tient pas compte des paquets à envoyer pour la gestion du protocole de transport ? Même question si on en tient compte.

c L'utilisateur bénéficie maintenant de l'architecture TCP/IP dans son ordinateur. Quel protocole de transport l'application utilise-t-elle ? Pourquoi ?

d TCP gère des segments de 64 Kilo-octets. Combien de segments différents TCP fabrique-t-il pour transporter la photo sur Internet ?

Solution

a La *T_SDU* est la photo à envoyer. Sa taille est de 448 000 octets. L'entité de Transport gère des unités de données de 1 Kilo-octet ; une *T_PDU* de données contient donc 1 000 octets de données, une *N_SDU* en contient 1 024.

L'entité de Réseau utilise des paquets de 128 octets, dont les 3 premiers représentent l'en-tête du paquet. Une *N_PDU* de données contient 125 octets, alors qu'une *L_SDU* contient 128 octets dans son champ de données.

b Nous avons vu que dans chaque paquet se trouvent 125 octets de données. Il faut : 3 584 paquets de données pour transporter la photo. En fait, il faut envoyer : 448 000/ 1 024 = 437,5 soit 437 messages de 1 024 octets et un message à moitié plein. Cela fait en tout 3 589 paquets, sans compter les paquets nécessaires à la gestion de la connexion de Transport.

c Il utilise TCP pour que le destinataire puisse recevoir les différents paquets de la photo dans l'ordre et sans erreur…

d TCP fabrique 6 segments de 64 Kilo-octets de données plus un dernier segment de données partiellement rempli.



Les réseaux locaux d'entreprise

Pour répondre à leurs besoins propres en informatique distribuée, les entreprises ont mis en œuvre des *réseaux locaux d'entreprise*, constitués d'un ou plusieurs réseaux locaux ou LAN (*Local Area Network*). Ils utilisent des protocoles simples car les distances couvertes sont courtes (de quelques centaines de mètres à quelques kilomètres) et les débits importants (jusqu'à plusieurs centaines de Mbit/s). Après avoir vu la normalisation des architectures de réseaux locaux, nous détaillerons les différentes techniques d'accès au support, spécifiques de ce type de réseau. Puis nous analyserons le fonctionnement des réseaux locaux de première génération pour mieux comprendre leurs évolutions technologiques. Nous verrons comment interconnecter ces différents réseaux, en insistant sur les commutateurs qui occupent une place de choix dans les réseaux actuels. Enfin, nous aborderons les réseaux sans fil.

1 Architectures de réseaux locaux

Les réseaux locaux informatiques répondent aux besoins de communication entre ordinateurs au sein d'une même entreprise. Il s'agit de relier un ensemble de ressources devant communiquer : stations de travail, imprimantes, disques de stockage, ordinateurs, équipements vidéo. Nés dans les années 1970, ils ont été proposés par les fournisseurs informatiques. Leur « simplicité » et leur popularité sont dues au fait qu'ils furent conçus pour des environnements privés, sans recours aux solutions normalisées que proposaient les opérateurs de télécommunications (qui se trouvaient en situation de monopole à cette époque). L'accès à Internet fut ensuite largement facilité, du fait que les équipements étaient reliés au sein de l'entreprise. Il n'y avait plus qu'à mettre en place un partage sécurisé de cet accès.

Des réseaux plus étendus ont prolongé les réseaux locaux (surtout aux États-Unis) : des réseaux métropolitains ou interurbains appelés MAN (*Metropolitan Area Network*) se sont développés pour relier les établissements d'une même ville. Les réseaux grande distance ou WAN (*Wide Area Network*) assurent l'interconnexion de tous ces réseaux aux niveaux national et mondial. Des mécanismes d'interconnexion permettent de relier les réseaux locaux aux autres types de réseaux.

Un réseau local se caractérise par des équipements géographiquement proches les uns des autres et qui coopèrent en utilisant le support de transmission pour diffuser les données : l'ensemble des autres équipements du réseau reçoit tout bit émis par un équipement du réseau local. Cette particularité est à la base des architectures spécifiques de réseaux locaux, standardisées dans les années 1980. La section suivante nous permet de découvrir l'organisation physique des réseaux locaux, l'adressage, la topologie, le câblage et la couche Liaison de données.

1.1 Standards IEEE

Le comité 802 de l'IEEE, essentiellement constitué de représentants des constructeurs américains, s'est occupé de l'architecture des réseaux locaux. Plusieurs documents définissent l'architecture proposée (voir figure 5.1) :

- Le standard 802.1 définit le contexte général des réseaux locaux informatiques.
- Le standard 802.2 définit la couche Liaison de données.
- Les standards 802.3, 802.4, 802.5 et 802.6 définissent différents protocoles d'accès au support, pour plusieurs types de supports physiques : paire métallique, câble coaxial ou fibre optique.
- Le standard 802.11 définit un protocole d'accès pour les réseaux locaux sans fil (WLAN, *Wireless LAN*).

D'autres standards ont vu le jour ultérieurement, au fur et à mesure de l'évolution technologique.

Par rapport au modèle OSI, l'architecture normalisée dans les réseaux locaux découpe la couche Liaison en deux sous-couches : MAC (*Medium Access Control*) et LLC (*Logical Link Control*). La première règle l'accès au support partagé. Elle filtre les trames reçues pour ne laisser passer que celles réellement destinées à l'équipement concerné. La seconde gère l'envoi des trames entre équipements, quelle que soit la technique d'accès au support. Les spécifications de l'IEEE ne concernent donc pas les couches situées au-dessus de LLC.



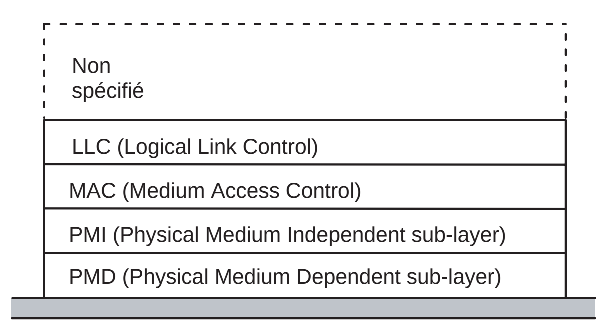


Figure 5.1

Modèle IEEE des réseaux locaux.

Comme on le voit à la figure 5.1, la couche physique est quelquefois découpée en deux niveaux : PMI (*Physical Medium Independent sub-layer*) qui assure le codage en ligne indépendamment du type de support de transmission utilisé, et PMD (*Physical Medium Dependent sub-layer*), qui s'occupe de l'émission physique du signal.

1.2 Adressage

Dans les réseaux locaux, l'adresse utilisée est une adresse physique qui se gère au niveau du matériel. Elle possède un format défini par l'IEEE sur 16 ou sur 48 bits. Ce dernier format constitue l'adressage universel des équipements : il correspond à un numéro de série dont un premier champ de 24 bits donne le constructeur de la carte (champ attribué par l'IEEE). Le second champ de 24 bits, librement choisi par le constructeur, est le numéro de la carte elle-même. De cette façon, toute carte réseau d'un ordinateur possède une adresse physique unique dans le monde[1]. Le format universel sur 48 bits est le plus utilisé (voir figure 5.2). Il est généralement baptisé *adresse MAC*, du nom de cette couche.

Figure 5.2

Format général des adresses MAC.

Adresse universelle sur 48 bits	
Adresse constructeur 24 bits	Adresse carte 24 bits

On peut également définir des adresses de groupe qui englobent plusieurs utilisateurs. Par exemple, dans le format universel, *l'adresse de diffusion* (ou *broadcast*) correspond à l'ensemble des équipements d'un réseau local. Dans cette adresse, tous les bits sont à 1. On l'écrit : `FF:FF:FF:FF:FF:FF` en hexadécimal.

Remarque

Les systèmes d'exploitation affichent l'adresse MAC de la carte réseau en hexadécimal, grâce à la commande `ifconfig` (pour Unix) ou `ipconfig` (pour Windows). On sépare les différents octets par deux points sous Unix ou par un tiret sous Windows, comme le montrent les deux exemples ci-après :
Sous Linux, la commande /sbin/ifconfig eth0 affiche (entre autres) :
eth0Link encap:EthernetHWaddr 00:90:27:6A:58:74
Sous Windows, la commande ipconfig /all affiche (entre autres) :
Adresse physique : 52–54–05-FD-DE-E5

1. Cette règle n'est plus vraiment respectée, car il est désormais possible de programmer soi-même l'adresse MAC de sa carte réseau (voir l'explication dans la remarque de l'exercice 6).

1.3 Topologie d'un réseau local

À partir des trois topologies de base : le *bus*, l'*anneau* et l'*étoile*, de nombreuses versions sont possibles. Il faut distinguer la *topologie physique* de la *topologie logique*. La première caractérise la manière dont est réalisé le câblage du réseau local (la structure des chemins de câbles, le type de raccordement…) ; la seconde décrit comment on attribue le droit à la parole entre toutes les stations. La topologie logique définit la *méthode d'accès au support* (ou *niveau MAC*) utilisée.

Topologie physique

La *topologie en bus* consiste à utiliser un long câble, sur lequel les différents équipements se raccordent en série, pour qu'il n'y ait qu'un seul chemin sans boucle entre deux équipements du réseau local. Chaque station peut accéder à tout moment au support commun pour émettre. Les données sont diffusées à toutes les stations. Le temps de propagation n'étant pas nul, il peut se produire des *collisions* lorsque différentes stations émettent au même moment. L'exemple type d'une topologie en bus est illustré figure 5.3. Cette topologie permet de faire des communications point à point et se prête naturellement à la diffusion. En revanche, toute coupure du bus entraîne une panne complète du réseau.

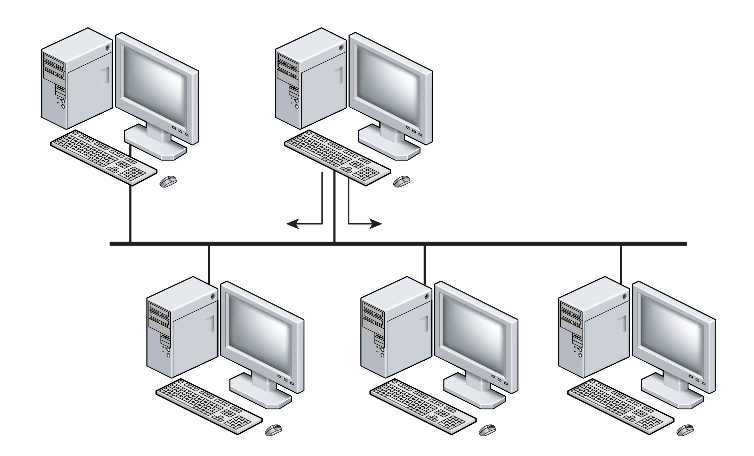


Figure 5.3
Topologie en bus.

Dans la *topologie en anneau*, chaque station est connectée au support par un port d'entrée et transmet les données à la station suivante par son port de sortie. Les différentes stations sont reliées en cascade et les données circulent d'une station à l'autre, toujours dans le même sens : chaque station traversée prend le message, l'analyse puis le retransmet sur son port de sortie (voir figure 5.4).

L'anneau manque de fiabilité en cas de rupture du support. On le double parfois pour réaliser deux anneaux qui peuvent transmettre soit dans le même sens soit en sens inverse. La seconde solution est préférable car elle permet de reconstituer le réseau, même en cas de rupture des deux anneaux au même endroit.

La *topologie en étoile* est, en fait, la généralisation des liaisons point à point : chaque équipement est relié par une liaison spécifique à un équipement central. La complexité de celui-ci dépend des modes de communication entre stations. Cette topologie présente un point faible : le réseau est inutilisable en cas de panne de l'équipement central, lequel peut constituer un goulet d'étranglement et entraîner la dégradation des performances du réseau s'il est mal dimensionné.

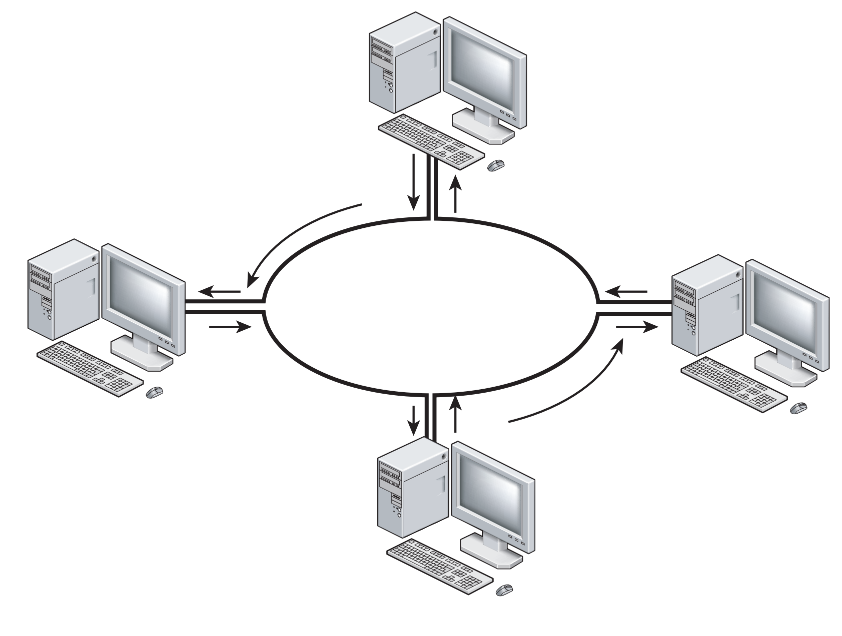


Figure 5.4

Topologie en anneau.

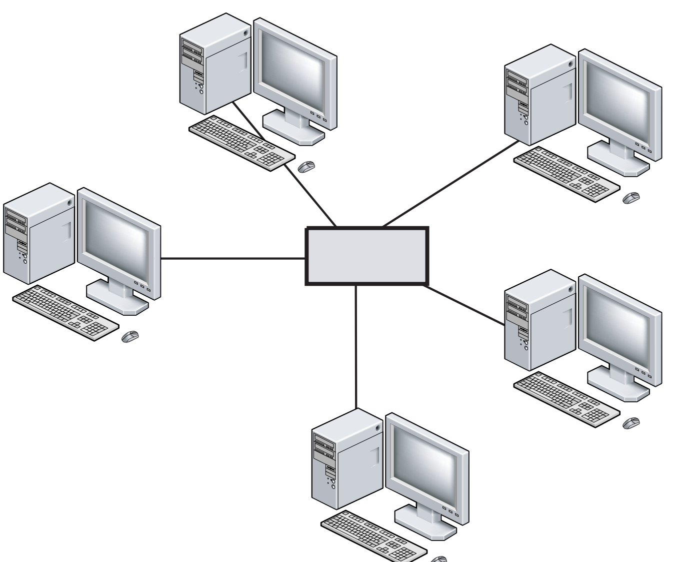


Figure 5.5

Topologie en étoile.

Topologie logique

La topologie logique s'appuie sur la manière dont les équipements échangent leurs données sur le réseau local. Elle ne dépend que du niveau MAC choisi et non de la façon de raccorder les équipements entre eux. Pratiquement, deux topologies logiques sont à considérer : le bus et l'anneau.

On peut en effet utiliser différentes topologies physiques pour réaliser une topologie logique donnée. Par exemple, une topologie logique en bus peut utiliser aussi bien un câblage physique en bus (cas du coaxial) qu'une topologie en étoile (pour un câblage physique par paires torsadées). De même, une topologie logique en anneau peut utiliser un anneau physique, un câblage en étoile autour d'un répartiteur, voire une topologie physique en bus !

1.4 POLITIQUE DE CÂBLAGE

La mise en place du câblage constitue un service de base, au même titre que l'infrastructure électrique des bâtiments. C'est pourquoi il faut disposer d'un système de câblage universel, adapté à la diversité des équipements et permettant la mise en œuvre de toutes les architectures de réseaux. Il existe deux possibilités de câblage : le *postcâblage* et le *précâblage*.

Le *postcâblage* consiste à installer l'infrastructure de communication, au fur et à mesure des besoins, dans des bâtiments qui n'avaient pas été prévus pour cela. On câble généralement le réseau local en calquant la topologie physique sur la topologie logique. L'accroissement du parc des équipements informatiques, les restructurations de sociétés, les déménagements donnent lieu à des modifications de câblage continuelles et coûteuses. Cette solution est de plus en plus obsolète.

Le *précâblage* se conçoit dès la construction du bâtiment. On le trouve aujourd'hui dans tous les bâtiments neufs, notamment dans les immeubles de bureaux. Il permet la mise en œuvre de toutes les topologies et consiste à poser une grande quantité de conducteurs offrant une grande souplesse d'arrangement. La présence des câbles est prévue à tous les étages, même si on ne connaît pas l'affectation future des locaux. Certains constructeurs proposent même une gestion technique du système de câblage. Le précâblage est évidemment moins coûteux pour l'entreprise.

1.5 COUCHE LLC

Le standard IEEE 802.2 définit un protocole de commande, LLC, fondé sur les principes du protocole normalisé HDLC que nous avons vu au chapitre 2. Trois classes sont définies :

- LLC1 fournit un service simple sans connexion ni contrôle, en point à point, en multipoint ou en diffusion.
- LLC2 assure un service avec connexion entre deux points d'accès et possède les fonctionnalités complètes du niveau Liaison du modèle OSI (contrôle de flux et contrôle d'erreur).
- LLC3, adapté au monde des réseaux industriels, rend un service sans connexion avec acquittement.

LLC1 est le protocole le plus courant dans les réseaux locaux informatiques. Il se réduit pratiquement à une seule trame : UI *(Unnumbered Information*), trame d'information non numérotée, correspondant à la notion de datagramme. Le service rendu par le protocole LLC1 est minimal : il se contente de formater les messages à émettre et de leur ajouter un bloc de contrôle d'erreur. Le récepteur vérifie le bloc de contrôle et détruit les messages reçus erronés. Il n'y a aucun accusé de réception, ni aucune demande de retransmission. Un tel fonctionnement est acceptable dans l'environnement des réseaux locaux car les distances ainsi que les taux d'erreur sont très faibles. Les messages manquants sont éventuellement détectés puis réémis au niveau de la couche Transport.

LLC2 est un protocole complet, analogue à la norme HDLC vue au chapitre 2. Quant à LLC3, il ajoute à LLC1 la notion d'accusé de réception. Dans les réseaux locaux industriels ou la commande de processus, il est important de garantir la fiabilité des transmissions, d'où l'idée d'un protocole sans connexion qui permette la bonne réception des messages sans la lourdeur imposée par la gestion des connexions.

Les trois classes de LLC étaient destinées à couvrir l'ensemble des besoins des utilisateurs. Aujourd'hui, la plupart des installations existantes se contentent de LLC1.



2 Techniques d'accès au support

Les réseaux locaux nécessitent un partage du support – donc de sa bande passante utile – entre les différents utilisateurs. Les constructeurs informatiques ont proposé de nombreuses techniques d'accès regroupées en deux grandes familles : les unes *à accès aléatoire*, les autres *à accès déterministe*.

Dans les techniques à accès aléatoire, chaque équipement émet ses données sans se soucier des besoins des autres. Plusieurs variantes sont fondées sur ce principe.

Dans les techniques déterministes, l'accès au support se fait à tour de rôle. L'accès est soit fixé *a priori* (indépendamment de l'activité des équipements), soit dynamiquement (en fonction de leur activité). Cette famille de techniques comprend tous les protocoles à *jetons*, dans lesquels le droit d'émettre est explicitement alloué à un équipement grâce à une trame particulière appelée *jeton*.

Remarque

Aloha, la plus ancienne méthode de contrôle d'accès à un support physique, appartient aux techniques aléatoires. Elle consiste à envoyer un message, sans s'occuper de ce que font les autres équipements. En cas de collision, le message est retransmis au bout d'un temps aléatoire. Son nom provient de l'archipel d'Hawaï car cette technique y fut expérimentée pour la première fois, dans un réseau hertzien reliant les différentes îles.

2.1. Techniques d'accès aléatoire

Les méthodes d'accès aléatoire portent le nom générique de CSMA (*Carrier Sense Multiple Access)*. Elles sont bien adaptées à la topologie en bus et exploitent la très faible distance entre les équipements. Quand un équipement a une trame à émettre, il se met à l'écoute du support[2], attend que celui-ci soit libre avant de commencer la transmission. Du fait des temps de propagation non nuls, un équipement peut provoquer une collision, même s'il a écouté au préalable et n'a rien entendu : plus le délai est grand et plus le risque de collision augmente.

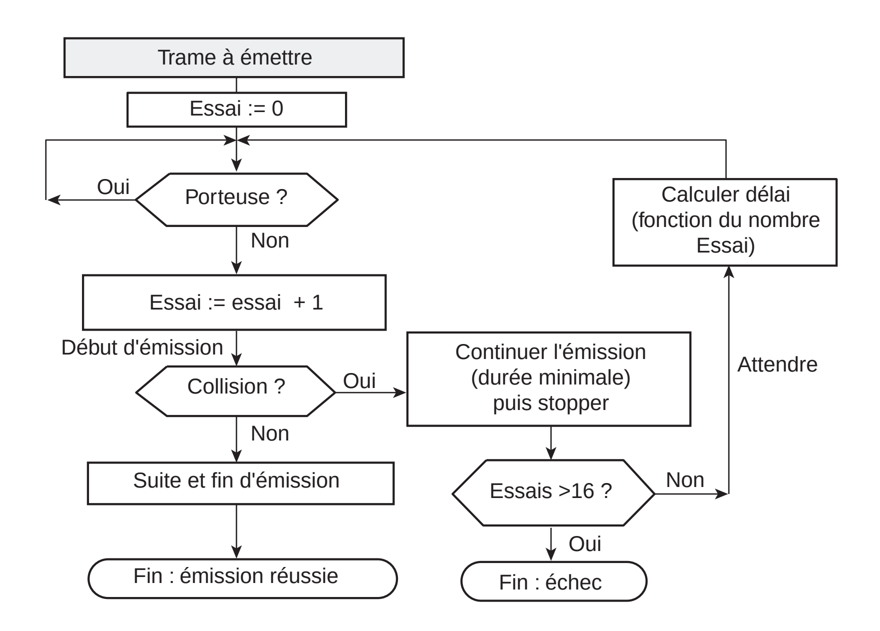
Il existe différentes variantes de ce mécanisme. La plus classique est normalisée sous le nom IEEE 802.3 : CSMA/CD (*CSMA with Collision Detection*). L'originalité de ce mécanisme, illustré à la figure 5.6, est que l'équipement *continue* d'écouter le support de transmission après le début de son émission. Il *arrête d'émettre*, après un très bref délai, s'il détecte une collision[3]. Le temps d'écoute pendant l'émission est limité à quelques microsecondes (il représente le temps de propagation aller et retour entre les deux stations les plus éloignées). La durée de la collision est ainsi réduite au strict minimum. La période pendant laquelle il est impossible d'éviter une collision malgré l'écoute préalable s'appelle *période de vulnérabilité*. La longueur maximale du bus détermine la durée maximale de cette période.

2. « Écouter » revient à mesurer la puissance du signal reçu : en effet, le rapport signal/bruit garantit qu'on sait faire la différence entre un signal de données et un simple bruit sur le support. Le support est libre si on ne détecte pas de signal transportant une donnée.

3. Lorsque deux stations émettent simultanément, leurs signaux se superposent et chaque émetteur ne reconnaît plus son message sur le support.

Figure 5.6

Mécanisme CSMA/CD.



Avec une technique aléatoire, le temps nécessaire pour émettre une trame ne peut être garanti. En effet, les retransmissions sont faites au bout d'un intervalle de temps qui dépend du nombre de tentatives. Après 16 tentatives infructueuses, l'équipement abandonne. L'intérêt de cette technique est sa simplicité de mise en œuvre, car elle ne nécessite pas la présence d'un équipement de contrôle. De plus, elle est totalement décentralisée, indépendante du nombre et de l'état des machines connectées.

2.2. Techniques d'accès déterministe

Les techniques déterministes utilisent un *jeton,* sur un bus ou sur un anneau. Le jeton est une trame qui circule dans le réseau d'équipement en équipement : un équipement *A* qui reçoit et reconnaît le jeton possède « le droit à la parole ». Il est autorisé à émettre sur le support (voir figure 5.7). Une fois sa transmission terminée, il transmet le jeton à l'équipement suivant. Le mode de transmission du jeton dépend de la topologie logique du réseau :

- Dans un anneau, l'équipement suivant est le premier équipement opérationnel, physiquement relié au précédent et en aval de celui-ci. La transmission du jeton (ou de toute trame) se fait toujours vers cet équipement, sans qu'il y ait besoin de le désigner explicitement : le jeton est *non adressé*.
- Dans un bus, l'équipement suivant est l'un des équipements du réseau, connu seulement du possesseur du jeton. Une trame contenant le jeton est diffusée sur le bus et possède l'adresse explicite du destinataire ou *successeur*. Chaque équipement n'a qu'un et un seul successeur dont il connaît l'adresse. On crée ainsi un *anneau virtuel* de circulation du jeton. Le jeton est *adressé*.

En fonctionnement normal, une phase de transfert de données alterne avec une phase de passation du jeton. Chaque équipement doit pouvoir traiter la réception et le passage du jeton, en respectant le délai maximal défini par la méthode d'accès. Il est également indispensable de prendre en compte l'ajout d'un nouvel équipement. Enfin, il faut réagir à l'altération, voire à la perte du jeton (cette trame, comme les autres, peut subir des erreurs de transmission) en mettant en place un mécanisme de *régénération* du jeton qui dépend du type du jeton (adressé ou non).



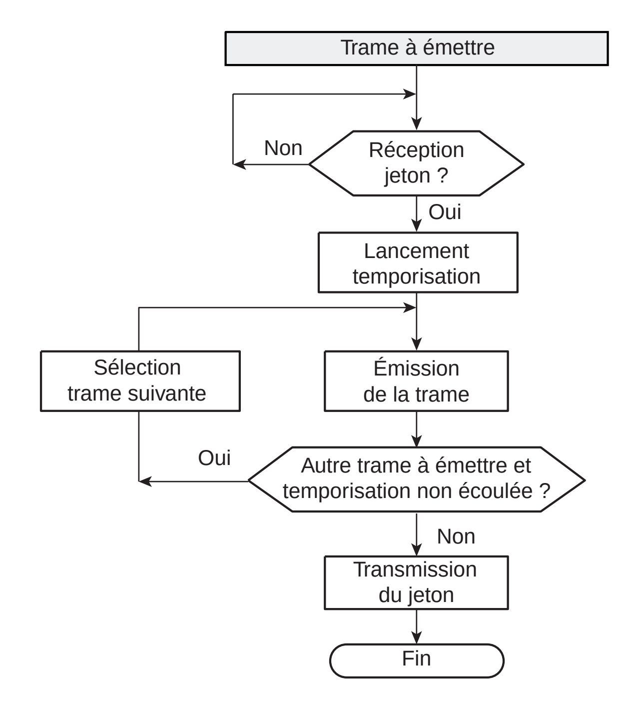


Figure 5.7

Mécanisme de jeton.

Pour mieux comprendre le fonctionnement des réseaux locaux, nous allons décrire les réseaux de première génération (*Ethernet*-IEEE 802.3 et *Token Ring*-IEEE 802.5). Ils diffèrent par leur organisation physique, leurs supports, leur plan de câblage, ainsi que par le format des trames. Nous verrons à la section 5 comment ces réseaux ont évolué au cours des trente dernières années.

3 Ethernet IEEE 802.3 de première génération

La société Xerox a développé Ethernet en 1976. Ce fut le premier produit de réseau local utilisant le mécanisme CSMA/CD sur un bus physique. Vu son grand succès, les sociétés Xerox, DEC et Intel ont décidé d'en faire un standard qui a servi de base au comité IEEE pour sa norme 802.3, même si Ethernet et le standard IEEE 802.3 diffèrent sur des points mineurs. La réussite d'Ethernet a été considérable : il est d'usage courant maintenant d'appeler Ethernet tout réseau local utilisant CSMA/CD, même s'il n'a plus grand-chose en commun avec le réseau initial.

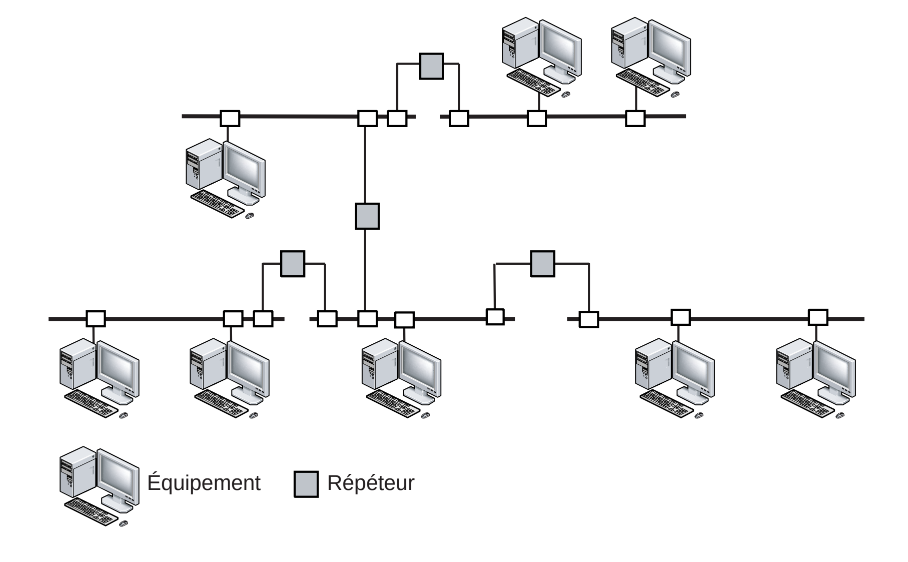
3.1 Organisation physique d'un réseau Ethernet

Les réseaux IEEE 802.3 utilisent une transmission en bande de base avec un code Manchester. Le réseau est organisé en un ou plusieurs segments, reliés de façon à conserver la structure de bus (voir figure 5.8). Afin que tous les équipements reçoivent un signal de puissance suffisante, la longueur de chaque segment est limitée. Pour des longueurs supérieures, il faut utiliser des *répéteurs*, qui décodent et amplifient les signaux reçus sans les interpréter. Ils contribuent à augmenter légèrement le délai de propagation et relient différents segments de façon à former un seul bus logique et un seul *domaine de collision* (ensemble des stations susceptibles de provoquer des collisions en cas d'émissions simultanées).

Pour limiter les risques de collision, le standard impose un délai de propagation aller et retour du signal strictement inférieur à 51,2 microsecondes.

Figure 5.8

Structure de bus « ramifié ».



> **Remarque**
>
> Chaque extrémité d'un bus est munie d'un *bouchon de terminaison* qui est, en fait, une résistance électrique dont l'impédance est égale à 50 Ω (impédance caractéristique du bus). Son rôle est d'absorber le signal électrique qui se propage, pour l'empêcher au maximum d'être réfléchi à l'extrémité du support et provoquer par là un brouillage du signal par lui-même. Le bouchon d'extrémité joue un rôle important dans la structure du réseau, puisqu'il absorbe littéralement le message émis sous la forme d'un courant électrique.

3.2 Format de la trame Ethernet

La figure 5.9. illustre le format de la trame Ethernet de base. Il comprend un long préambule (`101010`…) provoquant l'émission d'un signal rectangulaire de fréquence 10 MHz si le débit de transmission est de 10 Mbit/s. L'ensemble des équipements du réseau se synchronise ainsi sur le message émis. Le champ SFD (*Start Frame Delimitor)* contient la séquence `10101011` qui marque le début de la trame.

La trame contient dans son premier champ significatif l'adresse du destinataire DA (*Destination Address*) et celle de l'expéditeur SA (*Source Address*). Il s'agit des adresses MAC dont nous avons parlé à la section 1.2. Un champ sur deux octets précise la longueur (en nombre d'octets) des données de la couche LLC. La norme 802.3 ayant défini une longueur minimale de trame à 64 octets (qui représente à 10 Mbit/s un temps de transmission de 51,2 microsecondes), celle-ci est complétée par des octets de « bourrage » si la trame est plus courte. En fait, la taille de la trame doit être comprise entre 64 et 1 518 octets, ce qui laisse de 46 à 1 500 octets « utiles » dans le champ de données. La taille maximale est imposée pour assurer un rôle équitable entre les différents équipements (celui qui a réussi à prendre la parole ne peut pas la monopoliser…). La trame se termine par un champ FCS (*Frame Check Sequence).* Calculé par l'émetteur, le FCS permet au



récepteur de vérifier la validité des trames reçues. La détection des erreurs se fait à l'aide du polynôme générateur :

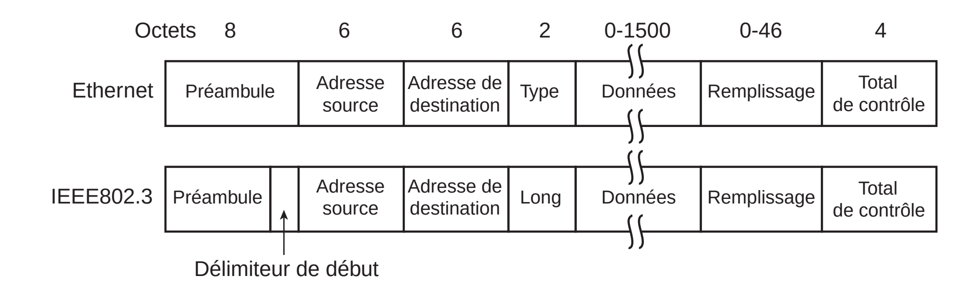
$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{5} + x^{4} + x^{2} + 1.$$

Une trame doit contenir obligatoirement un nombre entier d'octets. Enfin, un silence, obligatoire entre les trames, dure 9,6 microsecondes.

Figure 5.9

Format de la trame Ethernet.

(a) Ethernet
(b) IEEE802.3



Initialement, dans la norme IEEE 802.3, le champ longueur devait indiquer la longueur réelle du contenu de la trame. Dans la pratique, le contenu de la trame définit implicitement sa propre longueur. Ce champ, rebaptisé *type*, s'utilise désormais pour indiquer à quel protocole appartiennent les données encapsulées dans la trame. Par exemple, il peut prendre (en hexadécimal) les valeurs suivantes : `0800` (protocole IP), `0806` (protocole ARP), `0835` (protocole RARP). Nous reverrons au chapitre 6 le rôle de ces trois protocoles.

3.3 Supports et plan de câblage d'Ethernet

Historiquement, la première solution rencontrée est un plan de câblage en bus et le support utilisé un câble coaxial. Les équipements raccordés doivent respecter entre eux une contrainte de distance minimale. La nomenclature, sous la forme *XBase n*, décrit le débit du réseau et le support : *X* exprime le débit en Mbit/s, *Base* indique une transmission en bande de base, et *n* renseigne sur le type de câble. Les câblages initialement utilisés sont le *10 Base 5* et le *10 Base 2* :

- 10 Base 5 est un câble coaxial de 500 m maximum par segment, avec une transmission en bande de base et un débit de 10 Mbit/s. Il est à l'origine du produit Ethernet.
- 10 Base 2 est un câble coaxial plus fin donc plus maniable, de 180 m maximum par segment, avec une transmission en bande de base et un débit de 10 Mbit/s.

Le câble est posé dans des goulottes et alimente les différents bureaux. Le raccordement physique de la station au coaxial utilise une prise BNC (voir figure 5.10). Si le branchement d'un nouvel équipement est très facile à pratiquer, ce type de câblage présente toutefois deux inconvénients : la longueur maximale est facilement atteinte dans un bâtiment, et la coupure du bus empêche le fonctionnement du réseau.

Figure 5.10

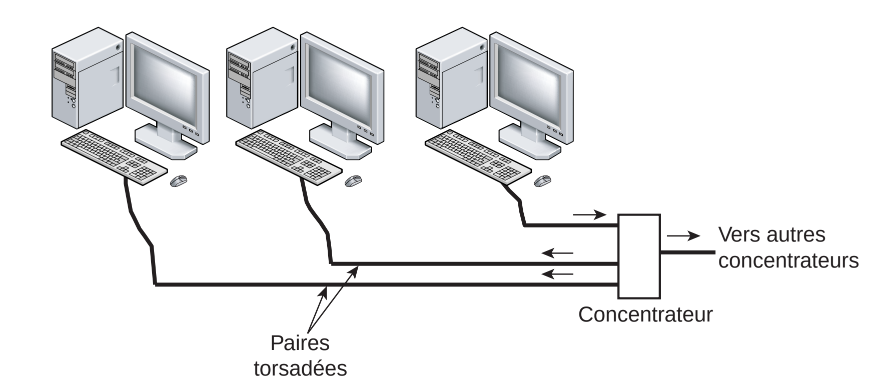
Connecteurs (a) BNC et (b) RJ45.



Dès les années 1990, on a recours au câblage en étoile (voir figure 5.11), dans lequel toutes les stations sont branchées sur un « concentrateur », ou *hub*, qui retransmet sur l'ensemble de ses ports tout signal reçu sur un port quelconque. La topologie logique reste celle d'un bus et le fonctionnement de l'accès par CSMA/CD est inchangé. Le support le plus courant fut alors la paire torsadée : 10 Base T (T pour *Twisted pair*) est une paire torsadée de 100 m par segment, transmettant en bande de base à un débit de 10 Mbit/s. La prise RJ45 remplace dans ce cas le connecteur BNC (voir figure 5.10). On peut aussi utiliser une fibre optique 10 Base F (*F* pour *Fiber*) de 2,5 km, transmettant en bande de base à 10 Mbit/s. Certains concentrateurs ont plusieurs ports pour raccorder des paires torsadées et un port pour raccorder une fibre optique, par exemple.

Figure 5.11

Câblage en étoile autour d'un concentrateur (hub).



Le concentrateur reste un équipement qui agit exclusivement au niveau du signal transmis : si la nature des supports change entre ses ports, il est simplement capable de récupérer les données binaires et d'en refaire le codage. Il n'interprète en aucun cas les données reçues.

3.4 Conclusion sur Ethernet

La grande force du standard IEEE 802.3 est sa simplicité : il n'y a aucun équipement centralisant le contrôle du réseau. L'ajout et le retrait d'un équipement se font sans interruption de fonctionnement, que ce soit avec un câblage en bus ou en étoile sur le concentrateur. Si le trafic est faible, l'accès au support est quasiment immédiat. En revanche, le réseau supporte mal les fortes charges qui peuvent provoquer un effondrement du débit utile, car le temps d'accès au support n'est pas borné. Un réseau 802.3 est donc une solution rapide et peu coûteuse à mettre en œuvre, destinée principalement à la bureautique. Les concentrateurs rassemblent en un point tous les raccordements physiques, ce qui améliore la sécurité et la rapidité d'intervention en cas de panne.

4 Token Ring IEEE 802.5

La société IBM a développé l'anneau à jeton ou Token Ring, standardisé par l'IEEE sous le nom 802.5. Les développements datent de la même époque qu'Ethernet mais les solutions proposées sont totalement différentes, tant dans l'organisation physique que dans le format des trames et les supports utilisés.



4.1 Organisation physique de l'anneau à jeton

La transmission se fait en bande de base avec un code Manchester différentiel (au lieu de coder chaque bit, le codage différentiel code la différence entre deux bits consécutifs). La topologie physique est un anneau simple unidirectionnel, dans lequel un équipement opérationnel actif sur l'anneau répète ce qu'il reçoit de l'amont vers l'équipement en aval. Un équipement en panne ou éteint ne participe pas à l'anneau (on dit qu'il est mis en *by-pass*) mais la propagation du signal est assurée. Des dispositifs électroniques ou électromagnétiques permettent à l'anneau de se reconfigurer automatiquement en cas d'incident.

À chaque instant, on distingue deux types d'équipements dans le réseau : celui qui « possède » le jeton et les autres. La topologie logique est un anneau dans lequel un équipement qui n'a pas le jeton se comporte comme un simple répéteur physique. L'équipement qui détient le jeton a le droit d'émettre une trame vers son successeur qui la transmet au suivant et ainsi de suite jusqu'à l'équipement émetteur. Celui-ci peut donc vérifier, en comparant la trame reçue avec la trame émise, que celle-ci a correctement fait le tour de l'anneau. Il peut savoir si le destinataire l'a correctement reçue et recopiée. Lorsqu'un équipement a fini de recevoir sa propre trame, il émet la trame spéciale contenant le jeton et repasse en fonctionnement de base.

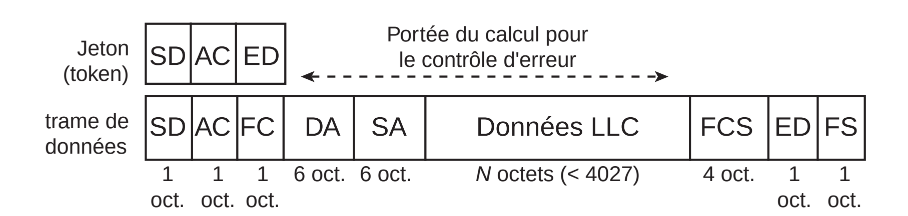
4.2 Format de la trame 802.5

La figure 5.12 illustre le format des trames 802.5. Lorsqu'il n'y a aucun trafic de données, le jeton circule dans l'anneau d'un équipement à l'autre. Il faut que la durée *t* entre l'émission d'un élément binaire et sa réception après un tour complet de l'anneau soit supérieure à la durée d'émission du jeton. On appelle *latence de l'anneau* la quantité d'informations qu'il contient à un instant donné. La latence doit être supérieure à la durée d'émission d'une trame de jeton codée sur 24 bits. Si l'anneau est trop court, l'équipement de surveillance ou *moniteur* (*Monitor*) gère une petite mémoire tampon pour retarder la répétition du signal et porter la latence à 24 bits.

Le champ SD (*Start Delimitor*) marque le début d'une trame. AC (*Access Control*) indique s'il s'agit d'une trame « jeton libre » ou d'une trame de données. En outre, cet octet contient un bit *M* géré par le moniteur et deux groupes de 3 bits, donnant respectivement la priorité du jeton (ou de la trame transmise) et la priorité des trames en attente dans les stations de l'anneau. FC (*Frame Control*) donne le type de la trame. Les champs d'adresses MAC (DA, SA) et le bloc de contrôle d'erreurs (FCS) sont définis comme dans IEEE 802.3. L'octet ED (*End Delimitor)* délimite la fin du jeton ou de la trame de données. Dans cette dernière, ED est suivi d'un octet FS (*Frame Status)* qui véhicule des informations de contrôle.

Figure 5.12

Format de la trame 802.5.



FS contient deux indicateurs (répétés par sécurité dans la seconde moitié de l'octet) : ARI (*Address Recognized Indicator*, ou indicateur d'adresse reconnue) et FCI (*Frame Copied Indicator*, ou indicateur de trame copiée). ARI est mis à 1 quand le récepteur reconnaît son adresse. FCI, quant à lui, est mis à 1 si le récepteur est parvenu à copier avec succès la trame provenant de l'anneau.

Les délimiteurs de début et de fin (SD et ED) sont des séquences particulières qui violent le principe du code Manchester : certains symboles de l'octet ne correspondent ni à un 0 ni à un 1 valides (on parle parfois de « non-données »).

4.3 Gestion de l'anneau

L'équipement détenteur du jeton peut émettre une trame qui fait le tour de l'anneau avant de lui revenir. Grâce aux différents indicateurs, l'équipement vérifie que l'anneau n'est pas coupé, qu'il n'y a qu'un seul moniteur actif et que le destinataire a bien copié la trame. Il détecte aussi la demande de jeton de plus haute priorité exprimée par un autre équipement du réseau. Après avoir reçu correctement sa propre trame, il émet un jeton libre sur l'anneau.

Pour éviter toute utilisation abusive du support, chaque station arme un temporisateur au début de la phase d'émission. Elle passe obligatoirement le jeton lorsque ce temporisateur expire, ce qui revient à déterminer la taille maximale d'une trame. On peut aussi affecter différentes priorités aux équipements du réseau. Celui qui a une trame en attente de priorité inférieure à celle du jeton ne peut prendre le jeton circulant. Il doit attendre le passage d'un jeton doté d'une priorité inférieure ou égale à celle de sa trame.

La mise hors service ou la panne de l'équipement qui possédait le jeton provoque la disparition de celui-ci. Un anneau à jeton est donc compliqué à surveiller : le moniteur crée un jeton à l'initialisation de l'anneau, surveille l'activité des équipements connectés, régénère le jeton en cas de perte, détecte les messages ayant fait plus d'un tour, assure la synchronisation bit, ajuste la latence de l'anneau, etc. En outre, pour remplacer le moniteur actif quand celui-ci tombe en panne, tous les autres équipements jouent le rôle de *moniteurs dormants* (*Standby Monitor*)…

Tant que le moniteur est opérationnel, il doit envoyer à intervalles réguliers une trame AMP (*Active Monitor Present*). Dès que cette trame n'est plus envoyée en temps voulu, une des stations dormantes émet une trame *Claim Token* pour prendre le contrôle de l'anneau. Si elle y parvient, elle devient le moniteur actif. Les stations dormantes signalent leur présence à intervalles réguliers en transmettant la trame SMP (*Standby Monitor Present*).

L'échange de trames AMP-SMP s'utilise également pour déterminer la liste des stations actuellement opérationnelles dans le réseau. Pour cela, le moniteur envoie une trame AMP et utilise l'adresse de diffusion générale (*broadcast address*) dans le champ adresse destination. Chaque station active de l'anneau propage le jeton libre engendré par la station la plus proche du moniteur. Elle émet une trame SMP contenant l'adresse de diffusion générale comme adresse destination et sa propre adresse comme adresse source (NAUN, *Nearest Active Upstream Neighbour*). La procédure se poursuit jusqu'à ce que toutes les stations actives de l'anneau aient répondu. À son retour dans le moniteur, la trame SMP contient l'adresse de son voisin aval, situé le dernier sur l'anneau.

Cette procédure est importante en cas de défaillance partielle ou totale de l'anneau. Si une station ne reçoit pas le flot de bits entrants, elle envoie une trame d'alarme *Beacon* pour signaler une condition d'erreur possible aux stations aval et au moniteur. Dans cette trame, la présence du champ NAUN facilite le diagnostic d'erreur.

4.4 Supports et plan de câblage

Le plan de câblage généralement proposé pour l'anneau à jeton est une étoile ou un ensemble d'étoiles. Un concentrateur actif AWC (*Active Wire ring Concentrator*) permet de constituer l'anneau (voir figure 5.13). Par des dispositifs électroniques ou électromécaniques, AWC surveille la présence active de chaque équipement (détection d'un équipement hors tension, d'un câble coupé…) et reconfigure l'anneau automatiquement en cas d'incident, en excluant l'équipement concerné (mise en *by-pass*). Il est possible de relier plusieurs concentrateurs entre eux pour augmenter la taille de l'anneau et le nombre des stations.

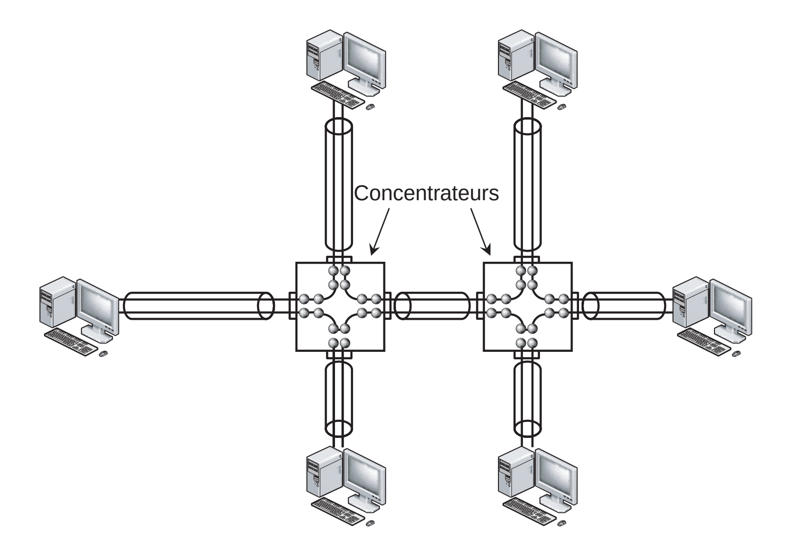


Figure 5.13

Câblage en étoile d'un anneau.

Le câble de raccordement entre l'équipement et le concentrateur est généralement une paire torsadée blindée d'impédance 150 Ω. Les débits possibles sont de 1 ou 4 ou 16 Mbit/s. Le nombre de stations dans l'anneau peut dépasser 200.

4.5 Conclusion sur l'anneau à jeton

Le débit utile d'un anneau résiste bien à la charge et ne s'effondre jamais comme avec la norme IEEE 802.3. Comme le délai d'accès au support est borné, on peut mettre en œuvre des dialogues entre équipements sur lesquels s'exécutent des applications temps réel. L'inconvénient principal de l'anneau à jeton réside dans la lourdeur et la complexité des mécanismes de sa gestion. Un tel réseau est donc globalement plus coûteux qu'un réseau Ethernet. Paradoxalement, les performances de l'anneau à jeton sont pénalisées à faible charge : le délai d'accès étant non nul, il faut attendre le jeton avant d'émettre alors que l'accès est immédiat en CSMA/CD sur un bus libre. De ce fait, l'anneau à jeton n'a pas pu offrir des débits supérieurs à 16 Mbit/s et n'a pu suivre l'accroissement des débits disponibles sur les réseaux Ethernet.

5 Évolution des réseaux locaux

Si Ethernet a été initialement conçu pour fonctionner sur des câbles coaxiaux à un débit de 10 Mbit/s, il est devenu le réseau local le plus répandu, dès qu'on a pu utiliser le câblage téléphonique et les paires métalliques. Deux évolutions majeures ont eu lieu simultanément :

l'utilisation de débits plus élevés et l'apparition des commutateurs. Enfin, l'avancée technologique a permis l'avènement des réseaux sans fil dont le développement est en plein essor, en raison du confort de raccordement qu'ils procurent.

5.1 FAST ETHERNET, ETHERNET COMMUTÉ, GIGABIT ETHERNET

Fast Ethernet est une version d'Ethernet à 100 Mbit/s compatible avec les réseaux à 10 Mbit/s. Elle a été largement diffusée dès le milieu des années 1990. Les concentrateurs proposés étaient bien souvent compatibles 10 et 100 Mbit/s. Ils se différenciaient simplement par leur nombre de ports. Gigabit Ethernet est la version à 1 Gbit/s (1 000 Mbit/s, standard 802.3z) qui a suivi. Les équipements Gigabit combinent généralement des ports à 10 et 100 Mbit/s avec une ou plusieurs connexions sur des fibres optiques à 1 Gbit/s. La paire métallique non blindée de catégorie 5 peut, elle aussi, supporter le débit de 1 Gbit/s sur de courtes distances. Une version Ethernet 10 Gbit/s est apparue en 2001 (Standard 802.3ae).

La fibre optique la plus utilisée est la fibre multimode. Dans ce support, un transducteur optique assure la transformation entre le signal lumineux et le signal électrique. La distance maximale entre deux équipements est de 1,5 km. Les nouvelles technologies issues des recherches les plus récentes promettent des fibres multifréquences (1 024 canaux par fibre) avec, pour chaque canal, un débit de plusieurs Go/s. Le principal désavantage de la fibre est son coût élevé.

Parallèlement, les concentrateurs ont été remplacés par des commutateurs (*switches*). Dans un réseau Ethernet commuté, tous les équipements du réseau sont reliés à un (ou plusieurs) commutateurs. La topologie physique peut être mixte : en étoile pour toutes les stations directement connectées au commutateur, en bus pour celles qui sont reliées *via* un concentrateur. Le commutateur, à la différence du concentrateur, lit les trames qu'il reçoit et exploite l'adresse du destinataire : il ne transmet la trame que sur le port qui permet d'atteindre le destinataire et non sur tous les ports. Si le port est occupé, le commutateur mémorise la trame et attend que ce dernier se libère. De plus, il possède des ressources de traitement élevées et peut gérer plusieurs trames simultanément. Il accroît donc énormément la capacité du réseau : par exemple au lieu de partager un débit de 100 Mbit/s entre tous les équipements reliés par un concentrateur, on obtient 100 Mbit/s dédiés à chacun d'entre eux dès lors qu'ils sont reliés par un commutateur : s'il y a 10 équipements dans le réseau dialoguant deux à deux, on peut obtenir un débit global de 500 Mbit/s.

Gigabit Ethernet s'est développé dans les environnements commutés et possède deux modes de fonctionnement : les modes *duplex intégral* et *semi-duplex*. Dans le mode duplex intégral, utilisé sur les liaisons point à point, un équipement émet et reçoit simultanément des données avec le commutateur ; il n'y a plus de collision possible. Le semi-duplex est employé pour les équipements raccordés par l'intermédiaire d'un concentrateur. Dans ce cas, des collisions peuvent encore se produire.

Grâce au débit employé, le temps d'émission d'une trame est très faible. Il a fallu apporter des fonctionnalités supplémentaires dans la méthode d'accès : l'*extension de trame* et le *mode rafale*. La première consiste à porter la longueur minimale de la trame à 512 octets (au lieu de 64 octets dans l'Ethernet classique) ; la seconde permet à un émetteur d'envoyer en une seule fois plusieurs trames consécutives. Ces deux fonctionnalités rendent supportable la contrainte de longueur maximale du réseau.

Il existe principalement deux technologies de commutateurs : *store and forward* et *cut through*. Quand un commutateur store and forward reçoit une trame, il la vérifie et, si elle ne possède pas d'erreurs, la stocke avant de l'envoyer sur le port adéquat. Ce fonctionnement convient bien au mode client/serveur car il élimine les trames erronées et accepte le

mélange de divers supports (cuivre-fibre optique, par exemple) ou encore le mélange de débits. Il présente l'inconvénient d'introduire un délai supplémentaire, puisque chaque trame est transmise deux fois. Un commutateur cut through analyse l'adresse MAC du destinataire et transmet la trame à la volée sans aucune vérification. Ce système fournit de faibles temps d'attente, mais il n'apporte aucun service à valeur ajoutée puisque même les trames incomplètes sont transférées. Une variante adaptative consiste à mesurer le taux d'erreur pendant le fonctionnement cut through et à basculer en store and forward si ce taux dépasse un certain seuil.

Enfin, les commutateurs peuvent intégrer des fonctions supplémentaires pour gérer, par exemple, une table de correspondance adresses MAC-numéros de ports sur plusieurs commutateurs reliés entre eux. On gère les commutateurs par une interface locale ou une interface Web.

5.2 Réseaux locaux virtuels ou VLAN (Virtual LAN)

L'introduction des commutateurs dans un réseau local a permis de construire des réseaux logiques, indépendants les uns des autres. Les réseaux sont désormais définis en fonction des centres d'intérêt de leurs utilisateurs, et non en fonction de la situation géographique des équipements au sein de l'entreprise. On parle alors de *réseaux locaux virtuels* ou VLAN (*Virtual LAN*).

Un réseau virtuel regroupe une communauté d'usagers répartis dans toute l'entreprise, comme s'ils appartenaient au même réseau physique. Les échanges à l'intérieur d'un VLAN sont sécurisés et les communications entre VLAN contrôlées. Par exemple, le réseau virtuel réservé à la direction de l'entreprise fournit un espace de communication sécurisé à l'équipe directoriale. Ce réseau est logiquement distinct du réseau virtuel affecté aux services de production, même si les machines des deux départements sont reliées physiquement aux mêmes commutateurs.

On utilise plusieurs techniques de différentiation des équipements pour créer un VLAN. La première opère au niveau des ports du commutateur : un sous-ensemble des ports correspond à un VLAN donné. Cette solution a l'inconvénient de ne pas gérer la mobilité des utilisateurs. La deuxième consiste à identifier les équipements d'un VLAN par leurs adresses MAC, quel que soit le port du commutateur sur lequel l'équipement est raccordé. Cette solution est plus souple que la précédente, mais elle lie encore l'appartenance à un VLAN particulier au matériel utilisé. La troisième utilise les adresses IP, nous la verrons au prochain chapitre.

Le commutateur contient une table de correspondance entre les VLAN et la liste des ports associés. Pour gérer le VLAN avec un maximum de souplesse (quelle que soit la technique de différentiation), il faut qu'il soit étiqueté (*tagged*), c'est-à-dire que les trames portent un identificateur du VLAN auquel elles appartiennent. Cette étiquette se résume par deux octets ajoutés dans la trame, selon les recommandations du comité 802 (standard 802.1Q). Nous évoquerons plus loin ce standard et ses évolutions.

6 Interconnexion des réseaux locaux

Physiquement, deux réseaux ne peuvent être reliés que par l'intermédiaire d'un équipement connecté à chacun d'eux, sachant acheminer des messages de l'un à l'autre. Plusieurs dispositifs d'interconnexion se mettent en place, selon le degré de similitude des réseaux :

l'équipement d'interconnexion peut être selon les cas un *répéteur*, un *pont*, un *routeur* ou une *passerelle* (voir figure 5.14).

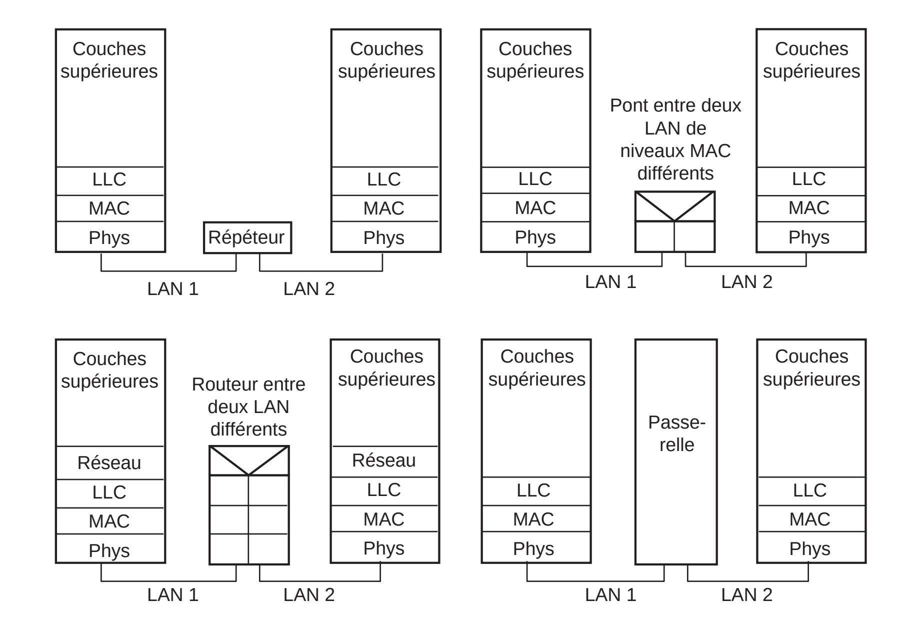


Figure 5.14

Répéteurs, ponts, routeurs et passerelles.

6.1 Répéteurs

Les *répéteurs* ne font que prolonger le support physique en amplifiant les signaux transmis. Ils propagent aussi les collisions. Ils sont utilisés pour relier deux segments de réseaux Ethernet, par exemple. Un répéteur n'a aucune fonction de conversion ou de transcodage. Il se contente de veiller à la répétition et à la régénération de signaux. Les répéteurs sont souvent utilisés pour s'affranchir des contraintes de distances préconisées dans les standards. Ils supposent donc que les architectures des sous-réseaux à relier soient identiques à partir de la couche MAC.

6.2 Ponts (*bridges*)

Les *ponts* (*bridges*) sont conçus pour construire un réseau local logique, à partir de plusieurs réseaux locaux, voisins ou distants. Ce sont des équipements qui interviennent au niveau de la couche LLC. Si les réseaux sont distants, deux demi-ponts peuvent être reliés par une liaison grande distance. Dans les deux cas, les réseaux reliés utilisent le même espace d'adressage MAC et constituent un réseau unique, les ponts étant transparents aux protocoles des couches supérieures. Les ponts améliorent les performances du réseau, dans la mesure où ils filtrent les collisions et ne les retransmettent pas. Ils ont évolué vers des équipements plus sophistiqués, comme les ponts *filtrants*, qui possèdent des fonctions particulières de sécurité et de contrôle du trafic : ils détectent, par exemple, les chemins redondants entre deux réseaux locaux grâce à un échange d'informations de gestion interne. L'algorithme exécute le protocole appelé STP (*Spanning Tree Protocol*), mis en œuvre pour éliminer le tronçon qui crée un chemin redondant et garder au réseau sa structure de bus ramifié.

Algorithme de l'arbre couvrant (*Spamming Tree*)

Cet algorithme, décrit dans le standard 802.1d[4], fait découvrir dynamiquement aux ponts un sous-ensemble sans boucle de la topologie du réseau. Pour cela, les ponts échangent des messages spéciaux permettant de calculer l'arbre couvrant. De tels messages sont



appelés *BPDU de configuration* (*Bridge Protocol Data Unit*). L'objectif des BPDU de configuration est de choisir :

- *Un pont unique de référence* (le pont *racine*). Ce pont sera considéré comme la racine de l'arbre parmi tous les ponts situés sur les réseaux locaux interconnectés.
- *Un pont dans chaque réseau local* (le pont *désigné*). Considéré comme le plus proche du pont racine, le pont désigné transmettra toutes les trames de ce réseau vers le pont racine.
- *Un port dans chaque pont* (le port *racine*). Ce port donne accès au meilleur trajet entre ce pont et le pont racine.
- *Les ports à inclure dans l'arbre couvrant*. Les ports qui composent l'arbre couvrant sont constitués du port racine, de tous les ports racine et de tous les ports où le pont est considéré comme pont désigné.

Le trafic des données est acheminé vers et en provenance des ports choisis pour faire partie de l'arbre couvrant. Jamais le pont ne retransmet de trames sur les ports n'en faisant pas partie : ces ports sont dans l'état bloqué.

Les BPDU sont transmises par un pont sur un port donné. Elles sont reçues par tous les ponts du réseau local rattaché au port et ne sont pas réexpédiées en dehors du réseau local. Dans une BPDU de configuration, l'adresse destination est une adresse spéciale attribuée à tous les ponts. L'adresse source est l'adresse physique associée au port : un pont possède autant d'adresses physiques que de ports. En outre, un pont possède un identificateur unique ID, codé sur 48 bits, qu'il utilise comme identificateur propre dans le champ de données d'un message de configuration. Nous utiliserons par la suite les termes :

- *ID racine*. Identification du pont supposé être la racine.
- *ID pont émetteur*. Identification du pont émettant le message de configuration.
- *Coût*. Coût du meilleur trajet depuis le pont émetteur jusqu'à la racine.
- *ID port*. Adresse physique d'un port.

À l'initialisation du protocole, chaque pont suppose qu'il est racine. Il émet donc des BPDU de configuration sur chaque port, avec son propre identificateur comme ID racine et ID pont émetteur et un coût nul vers la racine. Ensuite, il va recevoir continuellement des BPDU de configuration sur chaque port. Pour chacun d'eux, il sauvegarde le « meilleur » message de configuration, c'est-à-dire celui dont l'ID racine est le plus petit. En cas d'égalité d'ID racine, il choisit la BPDU dont le coût est le plus faible puis, si nécessaire, celle dont l'ID pont émetteur est le plus petit. Lorsque l'ID racine, le coût et l'ID pont émetteur sont identiques, c'est l'ID port qui sert d'arbitre. Une fois calculés la racine et le coût à la racine et après détermination du pont désigné sur chaque port, il faut décider quels ports doivent faire partie de l'arbre couvrant. Celui-ci est constitué du pont racine, de tous les ponts désignés et de tous les ports racine.

Exemple

Soit un pont d'ID = `92` qui a reçu un ensemble de BPDU de configuration conformément au tableau ci-après :

	ID racine	Coût	ID pont émetteur
Port 1	11	90	50
Port 2	11	83	41
Port 3	81	0	81
Port 4	17	32	26

4. Une version plus récente de cet algorithme est RSTP (*Rapid Spanning Tree Protocol*), décrit dans le standard 802.1w. Ce dernier protocole converge en quelques secondes au lieu d'une minute environ.

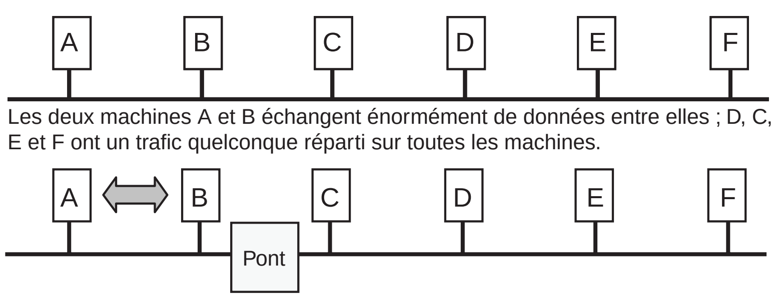
L'ID racine le plus petit est 11 ; le coût le plus faible parmi les messages ayant 11 comme ID racine est 83, donc le port 2 est le port racine. Le pont détermine ensuite sa distance au pont racine en comptant 83 + 1 soit 84 (dans la réalité, le nombre 1 est le résultat d'une mesure). La BPDU de configuration que peut émettre notre pont vaut : 11.84.92. Ce message est meilleur que celui qu'il a reçu sur les ports 1, 3 et 4. Le pont 92 est en conséquence pont désigné sur ces trois ports, sur lesquels il envoie sa BPDU de configuration.

Segmentation d'un réseau local

Les ponts permettent également de segmenter un réseau local en deux pour améliorer les performances. Par exemple, dans un réseau Ethernet qui approche de la saturation, on peut chercher les couples de machines qui ont un gros trafic entre elles et les isoler (voir figure 5.15). Le pont travaille par apprentissage : il apprend à situer les équipements progressivement, au fur et à mesure de leur activité. Dès qu'une trame se présente sur le pont et qu'elle est destinée au sous-réseau d'où elle vient, le pont la filtre (il ne la transmet pas dans un autre sous-réseau).

Figure 5.15

Segmentation d'un réseau local.



Un pont peut relier des réseaux locaux qui diffèrent par leur technique d'accès au support (un réseau utilisant CSMA avec un réseau utilisant des jetons, par exemple). Il doit alors gérer les différences de débit, de format, de méthodes d'accès et de services rendus. Le pont peut perdre des messages s'il est soumis pendant trop longtemps à des rafales de trafic sur l'un des réseaux qui dépassent la capacité de transmission sur l'autre. De plus, l'ensemble des différences nécessite un traitement dans le pont qui provoque un retard dans la transmission.

Grâce aux progrès technologiques, de nouveaux équipements, les commutateurs (*switches*), ont remplacé les ponts dans la plupart des installations. Ils prennent une place de plus en plus importante dans les réseaux d'entreprise car ils ont évolué et assurent désormais des fonctions plus sophistiquées que la simple commutation de trames.

6.3 Évolution des ponts : les commutateurs

L'essor des commutateurs a commencé à l'avènement des VLAN (que nous avons vu à la section 5.2). Le commutateur d'un réseau local peut être assimilé à un pont évolué à très hautes performances, qui transmet et filtre les trames grâce à ses tables de réacheminement. Dans les réseaux d'entreprise comptant plusieurs VLAN, les *trunks* sont des liaisons dédiées entre commutateurs, sur lesquelles circulent les données des différents VLAN.

Pour tenir compte des nouvelles topologies et des contraintes qu'elles ont imposées dans la circulation des flux d'information entre VLAN, certains protocoles antiboucles, comme



STP, ont été modifiés pendant que d'autres ont émergé : RSTP (*Rapid Spanning Tree*) ou 802.1w est la version modifiée de l'arbre couvrant qui permet une convergence plus rapide ; le MSTP (*Multiple Spanning Tree*), décrit dans le standard IEEE 802.1Q, permet de créer des arbres couvrants multiples pour les différents VLAN.

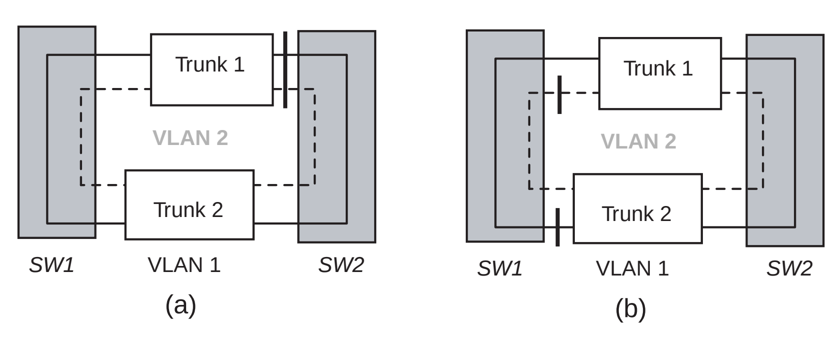
Arbre couvrant multiple dans les VLAN (802.1Q)

L'idée sous-jacente au concept de STP multiple est de proposer un algorithme qui tienne compte de la complexité de la circulation des flux des différents VLAN dans les trunks. Dans l'exemple de la figure 5.16a, deux commutateurs reliés par deux trunks transportent les données de deux VLAN (l'un des deux VLAN est en trait plein, l'autre en pointillé). La mise en œuvre d'un seul arbre couvrant conduit à bloquer un port dans chaque commutateur pour éviter les boucles. La figure 5.16b montre que l'utilisation de deux arbres couvrants évite ce problème.

Pour permettre une configuration dynamique des différents VLAN, les trunks doivent transporter les données de tous les VLAN.

Figure 5.16

Exemple de deux commutateurs reliés par deux trunks pour véhiculer les données de deux VLAN.



Configuration avec deux VLAN véhiculés sur deux trunks distincts entre les commutateurs SW1 et SW2. Un gros trait signifie qu'un port est bloqué.

(a) : Configuration utilisant un seul arbre couvrant. Pour éviter la création d'une boucle entre *SW1* et *SW2*, on ne peut pas se servir d'un des deux liens pour écouler le trafic normal des *VLAN*. Ici, *SW2* a ses deux ports bloqués : le second lien sert uniquement de secours en cas de panne du premier.

(b) : Configuration utilisant deux arbres couvrants. Dans ce cas, un des *trunks* transporte les données d'un *VLAN*, tandis que l'autre véhicule les données de l'autre *VLAN*. En cas de panne d'un *trunk*, le lien survivant peut transporter les données des deux *VLAN*.

Nous voyons bien que la contrepartie de la multiplication des VLAN dans le réseau de l'entreprise est la multiplication du nombre d'arbres couvrants à maintenir. Cette prolifération risque d'entraîner une gestion complexe de l'algorithme et provoquer une baisse des performances des commutateurs.

Pour rendre la circulation entre VLAN plus efficace, il s'est développé des techniques de *routage interVLAN*, naturellement assumées par les commutateurs, qui sont ainsi devenus des *commutateurs-routeurs*.

Commutateurs-routeurs

Les fonctionnalités de plus en plus étendues des commutateurs empiètent sur les fonctions classiquement dévolues aux routeurs. De ce fait, les commutateurs les plus sophistiqués sont souvent appelés des *commutateurs-routeurs*. Désormais, en plus des fonctions traditionnelles de commutation d'un port à l'autre, les commutateurs-routeurs sont capables d'effectuer des fonctions de niveau 3 et même de niveau 4 du modèle OSI.

Les fonctions de niveau 3 que peuvent exécuter les commutateurs-routeurs sont :

- Le routage interVLAN, en fonction des adresses IP.
- Le routage dynamique car ils peuvent exécuter les protocoles de routage comme RIP, OSPF, BGP… que nous verrons au chapitre 8.
- Le protocole VRRP (*Virtual Router Redundancy Protocol*), décrit par la RFC 2338. Ce protocole, de plus en plus utilisé – aussi bien dans les routeurs que dans les commutateurs-routeurs –, s'attache à résoudre le problème de l'unicité du routeur par défaut. Il est développé à la section suivante.
- La gestion de *listes de contrôle d'accès* ou ACL (*Access Control List*). Pour chaque sous-réseau IP, le commutateur peut autoriser ou interdire l'accès à tel autre sous-réseau IP, comme le fait normalement un routeur.

En plus des fonctions de niveau 3, les commutateurs-routeurs – comme la plupart des routeurs – peuvent inspecter le contenu des datagrammes IP. En effet, on peut affiner l'utilisation des listes de contrôle d'accès en autorisant ou en interdisant la circulation des flux de données sur certains ports TCP ou UDP. De la sorte, le commutateur-routeur se comporte comme un pare-feu de base décrit dans les compléments pédagogiques, sur le site www.pearsoneducation.fr.

Remarque

Ces nouvelles fonctions expliquent que les commutateurs sont des équipements d'interconnexion de plus en plus utilisés. Néanmoins, elles sont assurées en consommant des ressources (mémoire, processeur) utiles aux tâches normalement exécutées par les commutateurs : exécution du spanning tree et des autres protocoles antiboucles, apprentissage de la localisation des stations, gestion de la diffusion de niveau MAC, etc. En cas de trafic important, les performances du commutateur se dégradent si un grand nombre de listes de contrôle d'accès est mis en place.

Protocole VRRP (*Virtual Router Redundancy Protocol*) [RFC 2338]

Le protocole VRRP est un standard Internet qui propose une solution permettant à un réseau de ne pas être complètement isolé lorsqu'un équipement d'interconnexion de niveau 3 (routeur ou commutateur-routeur), unique dans le réseau, tombe en panne. Par exemple, si le routeur de sortie du réseau de l'entreprise existant en un seul exemplaire ne fonctionne plus, le réseau est complètement isolé du monde extérieur. VRRP décrit comment installer plusieurs équipements de secours qui prennent, automatiquement et en très peu de temps, la relève de l'équipement défaillant. Pour cela, deux routeurs (ou plus[5]) se partagent une adresse IP et une adresse MAC virtuelles ; un seul routeur est actif (*Master router*) à l'instant *t*. En cas de panne du routeur actif, le changement de routeur est transparent pour les utilisateurs.

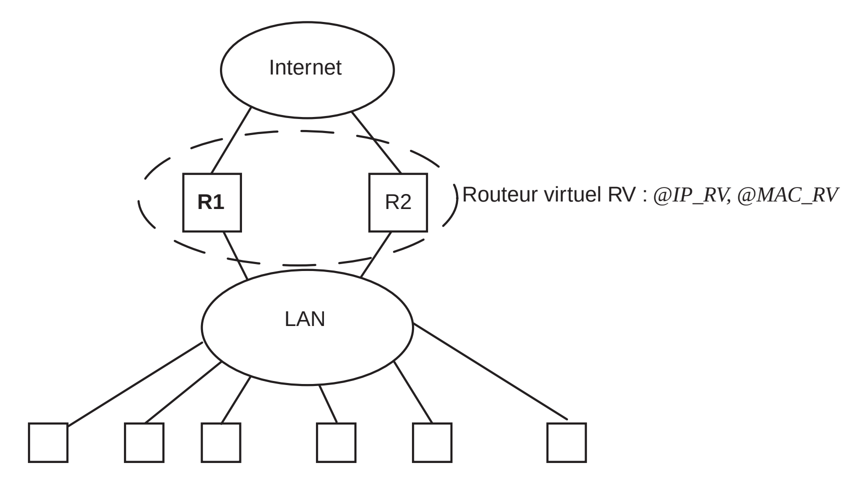
Les routeurs utilisant VRRP (les routeurs VRRP) se trouvent dans l'un des trois états suivants : *Initialize*, *Master* ou *Backup*. Dans l'état Initialize, le routeur attend un événement qui le fera basculer dans l'un des deux autres états. L'état Backup sert à vérifier que le routeur actif est bien dans l'état Master et qu'il est en fonctionnement. Dans l'état Master, le routeur actif informe les routeurs de secours (*Backup routers*) à intervalles réguliers qu'il peut toujours assurer le routage vers l'extérieur du réseau. La figure 5.17 donne un exemple d'utilisation de VRRP.

5. VRRP prévoit jusqu'à 255 équipements de secours mais la plupart des installations se contentent d'un seul équipement redondant.



Figure 5.17

Exemple de réseau utilisant le protocole VRRP.



R1 et *R2* forment le routeur virtuel. Ils possèdent leurs propres adresses *IP* et *MAC*. **R1** est le routeur actif. Les adresses réelles (*IP* et *MAC*) des routeurs sont respectivement : @IP_R1 ; @MAC_R1 pour le routeur R1 et @IP_R2 ; @MAC_R2 pour le routeur R2. Les machines du réseau ont comme seule adresse de routeur par défaut l'adresse du routeur virtuel, soit @IP_RV.

Les routeurs VRRP utilisent l'adresse MAC virtuelle : `00 00 5E 00 01 Id` du routeur VRRP (cet identifiant est codé sur un octet[6]). L'adresse IP virtuelle et les adresses IP réelles des routeurs sont déterminées par l'administrateur, en fonction de la structure du réseau.

Remarque

Il existe des variantes non standard de cet algorithme. On peut notamment citer le protocole HSRP (RFC 2281) de Cisco Systems Inc.

6.4 Routeurs (*routers*) et passerelles (*gateways*)

Les *routeurs* (*routers*) sont destinés à relier plusieurs réseaux de technologies différentes. Ils opèrent essentiellement au niveau de la couche 3 du modèle OSI, c'est-à-dire qu'ils assurent le routage des informations à travers l'ensemble des réseaux interconnectés. Le routeur possède au moins deux interfaces réseau et contient un logiciel très évolué, administrable à distance. Pour tenir compte de l'évolution des commutateurs, les routeurs proposent à leur tour des fonctions de niveau plus élevé que le niveau 3 : fonctions de pare-feu et autres, comme nous l'avons vu pour les commutateurs-routeurs. Ils sont liés à l'architecture des protocoles de routage utilisés, contrairement aux commutateurs. La majorité des routeurs utilisant le protocole IP, nous étudierons plus en détail leur fonctionnement au chapitre 6.

Enfin, les *passerelles* (*gateways*) sont des équipements qui relient des réseaux totalement différents : elles assurent une compatibilité au niveau des protocoles de couches hautes entre réseaux hétérogènes et effectuent, par exemple, des conversions vers des protocoles et des applications « propriétaires ». Notons que dans le jargon franglais des administrateurs de réseaux, le terme gateway désigne un routeur.

6. Les routeurs VRRP communiquent en multicast avec l'adresse `224.0.0.18`.

Remarque

Après avoir constaté l'évolution des commutateurs, on peut se demander dans ces conditions ce qui distingue réellement un commutateur-routeur d'un routeur…
Les routeurs ne se chargent pas de la gestion des VLAN (qui reste l'apanage des commutateurs), alors que les commutateurs ne gèrent pas de réseaux privés virtuels (VPN, *Virtual Private Network*[7]), pour lesquels les routeurs restent indispensables. En outre, le nombre de ports d'un commutateur est souvent beaucoup plus élevé que celui d'un routeur. Enfin, pour des fonctions de routage complexes, le routeur offrira de meilleures performances qu'un commutateur-routeur.

7 Réseaux locaux sans fil

Pour qu'une technologie puisse émerger, elle doit offrir, outre de nouvelles fonctionnalités, une certaine compatibilité avec des normes ou standards existants. Les contraintes qui ont guidé les concepteurs dans leurs choix techniques pour concevoir des réseaux sans fil étaient nombreuses : trouver une bande de fréquences disponible (de préférence mondiale) pour une grande diffusion des produits, tenir compte de la portée limitée des signaux radio, préserver la confidentialité des communications et de la durée de vie limitée des batteries des stations nomades, disposer d'une bande passante suffisante pour que le système soit viable économiquement et assurer une compatibilité ascendante. Le standard 802.11 pour réseaux locaux sans fil (WLAN, *Wireless LAN*) a été conçu pour être compatible avec Ethernet. De ce fait, les protocoles situés au-dessus de la couche MAC sont utilisés sans aucune modification.

Dans un WLAN, l'écoute préalable du signal avant émission ne fonctionne pas très bien, pour plusieurs raisons : par exemple, la disparité des puissances d'émission des différentes stations et la réflexion des ondes radio par des objets solides, entraînent des réceptions multiples du même message.

Après une brève description des standards de réseaux sans fil, nous présentons les techniques de transmission spécifiques de ces réseaux avant d'évoquer les différentes architectures : les réseaux *ad hoc* et les réseaux *à infrastructure*.

7.1 Standards des réseaux sans fil

On distingue deux grandes catégories de réseaux sans fil, selon leur usage et les performances attendues (voir tableaux 5.1 et 5.2) :

- réseaux sans fil (WLAN) compatibles Ethernet, standardisés par 802.11 ;
- réseaux sans fil (WPAN, *Wireless Personal Area Network*), reliant des assistants personnels (PDA), téléphones, etc. Standardisés par 802.15, ils sont plus connus sous le nom de Bluetooth.

Tableau 5.1
Normes WLAN

Normes WLAN	Nom commercial	Débit théorique en Mbit/s	Portée max
ETSI 300 652	Hiperlan1	20	–
ETSI (en cours)	Hiperlan2	54	30 m
802.11a	Wi-Fi	54	40 m
802.11b	Wi-Fi	11	90 m
802.11g	Wi-Fi	54	70 m
HomeRF 1.0	HomeRF	1,6	50 m

7. Les VPN sont présentés dans les compléments pédagogiques, sur le site www.pearsoneducation.fr.



Tableau 5.2

Normes WPAN

Normes WPAN	Nom commercial	Débit théorique en Mbit/s	Portée max
IrDA	FIR (Fast IR)	4	1 m
802.15.1	Bluetooth	1	30 m
802.15.3	Bluetooth 2	12	10 m
802.15.4	Zigbee	0,250	75 m

7.2 Techniques de transmission utilisées dans le standard 802.11

La bande de fréquences la plus utilisée pour les réseaux sans fil est dans la bande 2,4 GHz [2,4-2,4835 GHz]. Celle-ci est partagée par d'autres domaines d'applications (four à micro-ondes, transmetteurs domestiques, relais, télémesures, télémédecine, caméras sans fil…). Il y a donc des risques d'interférences ! Pour transmettre les données, les réseaux sans fil utilisent des combinaisons de modulations adaptées aux transmissions par radio (variantes de modulation de fréquence ou de phase) mais aussi des techniques spécifiques comme les techniques à étalement de spectre (*spread spectrum*) : elles utilisent une bande de fréquences large pour transmettre des données avec une faible puissance d'émission. La technique consiste à découper la large bande de fréquences en au moins 75 canaux de 1 MHz : dans la bande des 2,4 GHz, on peut ainsi créer 79 canaux de 1 MHz. La transmission s'effectue pendant environ 400 ms sur un canal puis sur un autre, en utilisant une combinaison de canaux connue de toutes les stations de la cellule.

Dans le standard 802.11b, la bande de 2,4 GHz est découpée en 14 canaux séparés de 5 MHz. Aux USA, seuls les 11 premiers canaux sont utilisables. En France, on n'utilise que les canaux 10 à 13. Pour transmettre correctement à 11 Mbit/s, il faut une largeur de bande de 22 MHz (théorème de Shannon). De ce fait, certains canaux recouvrent partiellement des canaux adjacents : il faut choisir des canaux isolés les uns des autres (par exemple, les canaux 1, 6 et 11). Dans la pratique, on utilise généralement des canaux distants de 25 MHz les uns des autres. Il faut donc organiser les points d'accès et l'utilisation des canaux pour éviter les interférences.

Dans le standard 802.11a, on utilise la bande des 5 GHz [5,15-5,35 GHz] et [5,725-5,825 GHz] et 8 canaux distincts, chacun ayant une largeur de 20 MHz.

7.3 Architectures des réseaux sans fil

Deux modèles d'architecture sont à considérer : les *réseaux ad hoc* et les *réseaux à infrastructure*. Dans les réseaux *ad hoc*, les communications s'effectuent en point à point entre les stations. C'est le modèle de fonctionnement des WPAN. Dans les réseaux à infrastructure, le réseau est géré par une ou plusieurs *bases* (ou *bornes* ou *points d'accès*). Lorsqu'un réseau comprend plusieurs bornes, celles-ci sont raccordées par un réseau Ethernet filaire. Chaque borne offre un ensemble de services appelés BSS (*Basic Service Set*). Les bases servent de ponts entre le réseau filaire et le réseau sans fil. Lorsqu'il existe plusieurs bornes, il faut mettre en place un service étendu afin de permettre aux utilisateurs de se déplacer d'une base à l'autre. L'ensemble des bornes constitue le *système de distribution*. Outre l'acheminement des données, les services fournis par un système de distribution sont :

- *L'authentification* (pour ajouter une station dans le réseau). Elle se fait le plus souvent par l'adresse MAC. La *désauthentification* est le service opposé au précédent qui gère correctement la sortie d'une station du WLAN.

- *L'association*. Elle permet à une station d'échanger des données *via* un point d'accès auprès duquel elle s'est identifiée. La *réassociation* permet d'aller d'une base à l'autre tandis que la *désassociation* permet de quitter une base ou le WLAN.
- La *confidentialité.* Cela consiste à utiliser une méthode de chiffrement.
- La *distribution.* C'est l'équivalent du routage dans un réseau classique.

7.4 Méthode d'accès dans les WLAN

802.11 utilise CSMA/CA (*Collision Avoidance*) pour gérer les contentions d'accès à la fréquence partagée par toutes les stations d'une base. Une station n'émet que si elle ne détecte pas de trafic sur la bande de fréquences partagée. Sinon, elle attend un temps aléatoire avant de se remettre à l'écoute. Pour minimiser les collisions, on utilise souvent un mécanisme optionnel : avant de lui envoyer une trame, la base envoie d'abord à la station une trame RTS (*Request To Send*), à laquelle celle-ci doit répondre et attendre ensuite la réception de la trame de données. Les autres stations, qui détectent la trame RTS, retardent leur éventuelle émission.

Contrairement à Ethernet, les récepteurs doivent envoyer une trame d'acquittement (ACK) pour chaque trame d'informations reçue, car les fréquences radio peuvent être perturbées. De plus, pour minimiser l'impact des interférences, les stations échangent des trames courtes.

Les stations doivent pouvoir passer d'une base à l'autre sans que la communication soit coupée (*roaming*). La station, identifiée auprès de plusieurs bases, détermine la meilleure (celle qui lui offre la meilleure qualité de transmission), avec laquelle elle doit être en contact, et se réassocie avec.

Remarque

Le nom BSS est parfois synonyme de borne dans la terminologie des réseaux sans fil.

Résumé

L'utilisation d'un support unique partagé entre plusieurs utilisateurs d'un réseau local nécessite la mise en œuvre de méthodes d'accès spécifiques (accès aléatoire avec détection de porteuse ou mécanismes à jetons). Par ailleurs, les réseaux locaux permettent la diffusion de l'information dans tout le réseau. Grâce à sa simplicité et sa capacité d'adaptation, Ethernet est le réseau le plus répandu. Depuis les origines, il a su évoluer du réseau en bus à 10 Mbit/s jusqu'au réseau en étoile autour d'un commutateur pouvant gérer des réseaux locaux virtuels avec des débits dépassant le Gbit/s. Selon le niveau de l'interconnexion, les réseaux locaux se relient au monde extérieur par différents équipements : répéteurs, ponts, commutateurs, commutateurs-routeurs, routeurs et passerelles. En outre, nous avons présenté les particularités des réseaux locaux sans fil.



Problèmes et exercices

Exercice 1 Câbler un petit réseau local à la maison

Énoncé

Comme vous possédez plusieurs ordinateurs à la maison (3 PC et 2 portables), vous souhaitez les mettre en réseau de la manière la plus simple possible.

a Énumérez les problèmes que vous devez résoudre pour mener à bien votre installation.

b Pour éviter de tirer trop de câbles dans la maison, vous décidez de relier les machines par des liaisons sans fil. Quelles sont les conséquences sur votre installation ?

Solution

a Il faut tout d'abord disposer des matériels et des logiciels appropriés. Pour cela, vous devez choisir le réseau local que vous voulez créer (Ethernet ou anneau à jeton), et la topologie physique que vous allez utiliser. Vous optez pour des cartes Ethernet, afin de créer un réseau local plus simple et moins coûteux à installer. Vous devez ensuite décider comment raccorder vos ordinateurs : topologie physique en bus ou en étoile ?

La topologie en bus est la solution la plus économique si vos ordinateurs sont situés dans la même pièce. La topologie en étoile, désormais la plus populaire, impose l'achat d'un concentrateur (hub) dont le prix dépend du nombre de ports disponibles. Cette dernière solution vous permettra de faire évoluer plus aisément votre installation (mais aurez-vous plus d'une dizaine de machines à la maison ?).

Vous décidez donc de raccorder vos machines en bus. Les étapes de votre installation sont : achat et assemblage des différents matériels, installation des logiciels, configuration des adresses IP.

Au terme de la première étape, vous devez posséder les matériels suivants :

- un câble dit « Ethernet fin » ;
- autant de prises BNC en T que vous raccordez d'ordinateurs sur le câble ;
- des prises BNC femelles pour raccorder les prises précédentes sur le câble ;
- des bouchons de terminaison aux extrémités du câble ;
- des cartes réseau (ou cartes Ethernet), une par ordinateur à connecter. Pour les portables, vous choisissez plutôt des cartes équipées de deux connecteurs (un connecteur BNC et un connecteur RJ45), pour pouvoir utiliser la même carte si vous changez de réseau physique. Vous pouvez vous contenter de cartes avec un connecteur BNC pour les autres machines.

Vous devez également disposer, sur chaque machine connectée, des logiciels de communication :

- un pilote (*driver*) pour chaque carte réseau, en général fourni par le constructeur de la carte ;
- une pile TCP/IP par ordinateur, le plus souvent fournie avec le système d'exploitation de votre machine ;
- un navigateur par ordinateur.

Il vous reste à tout assembler pour achever la deuxième étape ! Pour la troisième étape, les systèmes d'exploitation modernes possèdent souvent des fonctions de type *Plug and Play* (littéralement : branchez et jouez) ; les pilotes et autres logiciels sont alors très faciles à installer. Reste la dernière étape : l'affectation des adresses IP à toutes les machines. Cette étape sera vue au chapitre 6 qui traite du protocole IP.

b La conséquence immédiate de ce choix est que toute votre belle installation est à jeter ! Si vous souhaitez installer le réseau sans fil le plus simple qui soit, vous équipez tous les ordinateurs avec une carte Wi-Fi au lieu de la carte réseau précédente. Toutes les applications (partage de l'imprimante, jeux en réseau…) qui utilisent la pile TCP/IP seront utilisables sur vos machines. Cette architecture est une architecture *ad hoc*, décrite dans le standard 802.11.

Exercice 2 Différences entre 802.3 et 802.5

Énoncé

a Pourquoi la trame IEEE 802.3 (Ethernet) ne contient-elle pas de fanion de fin comme une trame type HDLC ?

b Pourquoi la trame IEEE 802.5 (Token Ring) ne contient-elle pas un long préambule comme la trame IEEE 802.3 ?

Solution

a La trame Ethernet 802.3 ne contient pas de fanion de fin car elle est suivie d'un silence obligatoire (intervalle intertrame), et sa longueur est codée dans le champ longueur. Dans le cas où le champ longueur est remplacé par un champ *type*, il faut extraire la longueur du contenu lui-même.

b Avec Ethernet, n'importe quelle station peut à un moment donné prétendre prendre la parole. Pour une station qui reçoit, l'émetteur est inconnu et se situe à une distance quelconque, variable d'une transmission à la suivante : il est nécessaire de refaire la synchronisation à chaque réception de trame. Avec Token Ring, une station reçoit toujours les données de son prédécesseur sur l'anneau. La synchronisation est donc beaucoup plus simple à acquérir.

Exercice 3 Bouchon de terminaison

Énoncé

Que se passe-t-il dans un réseau local en bus s'il n'y a pas de bouchon de terminaison ?

Solution

Aucune transmission n'est possible. Le bouchon a un rôle électrique, il doit avoir une impédance bien adaptée de telle sorte que les signaux ne soient pas réfléchis en arrivant aux extrémités du câble. La réflexion est une source de bruit qui perturbe toutes les transmissions.

Exercice 4 Période de vulnérabilité

Énoncé

Soit un réseau Ethernet en bus de 8 stations. La distance moyenne entre stations est de 15 m. La vitesse de propagation est de 250 m/µs. Quelle est la durée de la période de vulnérabilité ?

Solution

Si les stations sont réparties tous les 15 m, la distance entre les deux stations les plus éloignées l'une de l'autre est de 15*7 = 105 m. La période de vulnérabilité correspond au temps de propagation aller et retour entre les deux stations les plus éloignées soit :

$$2*105 / 250 = 0,84\ \mu s.$$



Remarque

Sur un bus aussi court, la probabilité qu'il y ait une collision est très faible : il faudrait que deux (ou plusieurs) équipements aient écouté et pris la décision d'émettre dans le même intervalle de 0,84 µs. D'où l'intérêt d'utiliser des bus plutôt courts.

EXERCICE 5 LONGUEUR ÉQUIVALENTE D'UN BIT

Énoncé

Dans un réseau local dont le débit binaire est de 5 Mbit/s et la longueur de 1 km, les signaux se propagent à la vitesse de 250 m/µs. À quelle longueur de câble correspond un bit transmis ? Cela a-t-il une influence sur le choix de la taille des messages ?

Solution

Si le débit est de 5 Mbit/s, un bit dure $1/(5*10^6) = 0,2$ µs, soit avec la vitesse de propagation de 250 m/µs, une longueur équivalente à 50 m de câble. Dans le réseau local dont la longueur est 1 km, soit 1 000 m, cela suppose qu'il y ait, à un instant donné, 1 000/50 = 20 bits. Cette longueur est donc très petite : le message est à la fois en cours de transmission et en cours de réception.

Remarque

Dans un réseau local dont le débit n'est pas très élevé, il est inutile de prévoir des protocoles complexes avec anticipation : à un instant donné, il n'y a qu'un (et un seul) message en cours d'émission.

EXERCICE 6 ADRESSE MAC

Énoncé

Une entreprise dispose d'un réseau Ethernet. Un nouvel employé dans l'entreprise est doté d'un ordinateur ayant une carte Ethernet d'adresse universelle `3E:98:4A:51:49:76` en hexadécimal. À quel niveau cette adresse est-elle gérée ? Est-il nécessaire de vérifier qu'aucun autre ordinateur ne dispose de la même adresse dans le réseau local ?

Solution

L'adresse MAC est l'adresse physique de la carte Ethernet. C'est le numéro de série de cette carte, défini par le constructeur de la carte. Les constructeurs ont des préfixes uniques au monde (3 octets) et numérotent ensuite leurs cartes sur les 3 octets suivants : deux cartes ne peuvent jamais avoir le même numéro de série. Il est donc impossible qu'un autre ordinateur possède la même adresse.

Remarque

Pour simplifier le travail des administrateurs responsables du parc de machines, il est possible de flasher la PROM qui contient l'adresse MAC. Bien que cette technique viole la règle d'unicité des adresses MAC au sein d'un réseau donné, elle évite la mise à jour les tables de correspondance entre adresses MAC et adresses IP en cas de remplacement d'une carte réseau défectueuse, par exemple.

EXERCICE 7 DÉBIT UTILE THÉORIQUE

Énoncé

On rappelle que le débit nominal d'un réseau Ethernet est de 10 Mbit/s et que les trames contiennent un préambule de 8 octets, deux champs d'adresse de 6 octets chacun, un champ longueur de 2 octets, des données dont la longueur est obligatoirement comprise entre 46 et 1 500 octets et un bloc de contrôle d'erreur de 4 octets. Par ailleurs, un intervalle de silence entre trames est obligatoire : sa durée est de 9,6 µs.

a Déterminez le débit utile maximal sur un réseau Ethernet. Que pensez-vous du résultat obtenu ? Pourquoi ne peut-on pas l'atteindre ?

b Quel est le degré du polynôme générateur utilisé pour le contrôle d'erreur ?

Solution

a Le débit utile maximal est obtenu de manière théorique si une station unique émet en permanence (en respectant l'espace intertrame) des trames de longueur maximale. On obtient alors :

Longueur totale équivalente d'une trame en octets = 8 (préambule) + 6 (adresse destinataire) + 6 (adresse émetteur) + 2 (longueur ou type) + 1 500 (contenu utile) + 4 (bloc de contrôle d'erreurs) + 12 (correspondant au silence intertrame) = 1 528 octets.

Le débit utile vaut = 10*(1 500/1 528) = 9,82 Mbit/s soit un rendement de 98,2 %.

Cela est bien évidemment un calcul théorique : il est impossible d'atteindre un tel rendement dans la pratique, dès que plusieurs équipements tentent d'émettre. Il y aura des silences et des collisions qui entraîneront d'éventuels silences et/ou collisions supplémentaires.

b Le bloc de contrôle d'erreur a une longueur de 4 octets soit 32 bits. Donc le polynôme générateur utilisé est de degré 32.

Remarque

En pratique, on considère qu'un rendement de 50 à 60 % est une valeur limite. Si le trafic devait être plus important, les performances s'effondreraient. Cet exercice montre l'intérêt des ponts et des commutateurs pour segmenter les réseaux locaux.

EXERCICE 8 DÉBIT UTILE RÉEL

Énoncé

Un réseau local en bus de type 802.3 a un débit de 10 Mbit/s et mesure 800 m. La vitesse de propagation des signaux est de 200 m/µs. Les trames MAC contiennent 256 bits en tout. L'intervalle de temps qui suit immédiatement une transmission de données est réservé à l'émission de l'accusé de réception de 32 bits.

a Quel est le nombre de bits en transit sur le bus à un instant déterminé ?

b Quel est le débit utile réel du réseau, en supposant qu'il y ait 48 bits de service (champs MAC et LLC) dans chaque trame ?

Solution

a Si le débit est de 10 Mbit/s, un bit dure $1/(10*10^6) = 0{,}1$ µs soit, avec la vitesse de propagation de 200 m/µs, un temps correspondant au parcours dans 20 m de câble. Dans le réseau local dont la longueur est 800 m, cela suppose qu'il y ait, à un instant donné, 800/20 = 40 bits.

b Le temps total pour transmettre une trame et son accusé de réception est de $(256 + 32)/(10*10^6) + 2*800/(200*10^6) = 28{,}8 + 8 = 36{,}8$ µs.



Dans ce calcul, nous comptabilisons le temps de transmission d'une trame de 256 bits, plus son accusé de réception (soit 32 bits), plus un temps de propagation aller et retour en prenant les équipements à distance maximale. Les informations utiles dans la trame sont de 256 – 48 = 208 bits. Il faut donc 36,8 µs pour transmettre 208 bits utiles. Le débit utile est de 208/36,8 = 5,65 Mbit/s.

EXERCICE 9 TAILLE MINIMALE DES TRAMES ETHERNET

Énoncé

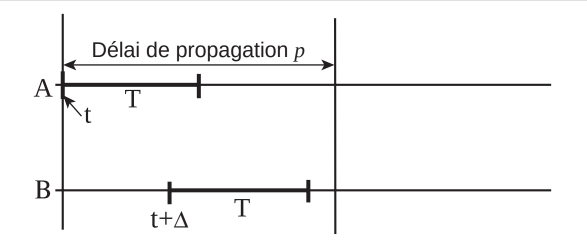
On considère un réseau local en bus utilisant le mécanisme CSMA/CD. On appelle *A* et *B* les deux équipements les plus éloignés. On note :

- *p*, le temps de propagation entre les stations *A* et *B*, *T* le temps de transmission d'une trame (toutes les trames émises sont supposées avoir la même longueur). Par hypothèse $T < p$.
- Δ, le temps séparant les débuts d'émission des stations *A* et *B*. Par hypothèse, on prend également $\Delta < p$.

La station *A* émet une trame à l'instant initial *t* et la station *B* à $t + \Delta$, comme le montre la figure 5.18.

Figure 5.1

Délai de propagation et temps de transmission d'une trame.



a Les deux stations peuvent-elles détecter la collision ?

b En déduire la taille *M*, exprimée en octets, du message de longueur minimale pour que toutes les stations puissent détecter une collision.

c Pourquoi le standard 802.3 impose-t-il un nombre maximal de répéteurs à traverser entre deux stations d'un même réseau d'entreprise utilisant des réseaux Ethernet ?

Solution

a Pour que toutes les stations détectent la collision, il faut qu'on ait $T = p + \Delta$, qu'on peut borner supérieurement par $T = 2\,p$. Dans ce cas, aucune station ne détecte la collison.

b Puisque $T = M/8\Delta$, on trouve $M = 16\,p\Delta$, en remplaçant *T* par sa valeur dans l'expression ci-dessus.

c Les répéteurs introduisent un délai supplémentaire, ils interviennent donc dans la valeur de *p*.

Remarque

On comprend pourquoi la norme 802.3 impose une taille minimale pour les messages émis par les équipements d'un réseau local de type CSMA/CD. Les récepteurs font ensuite le tri entre les « résidus de collision » trop courts et les « vraies » trames d'une longueur suffisante.

Énoncé

Soit un réseau local en bus utilisant un protocole de type CSMA/CD et comptant 4 stations notées *A*, *B*, *C* et *D*. Le temps est découpé en intervalles notés ST (*Slot-Time*), de durée égale à 51,2 µs.

On supposera que toutes les trames sont de longueur fixe et que la durée d'émission d'une trame quelconque est de 6 ST. À l'instant $t = 0$, la station *A* commence à transmettre une trame. À $t = 2$ ST, les stations *B* et *C* décident chacune de transmettre une trame et à $t = 5$ ST, la station *D* décide de transmettre une trame. On suppose que lors d'une collision, les deux machines impliquées interrompent leur communication et attendent un délai aléatoire avant de réémettre. La valeur de ce délai (exprimé en nombre entier de ST) est déterminée par l'algorithme suivant : après la première collision, une machine attend un temps aléatoire, égal soit à 0 soit à 1 ST ; après la deuxième collision, elle attend un temps aléatoire uniformément réparti entre 0 et 3 ST ; après *i* collisions, elle attend un temps aléatoire uniformément réparti entre 0 et $2^i - 1$ ST (si *i* est inférieur ou égal à 10) et entre 0 et 1 023 ST si *i* est compris entre 11 et 16. Au-delà de 16 collisions, elle abandonne la transmission.

On néglige le délai intertrame (on suppose donc qu'une trame peut être émise par une machine dès que celle-ci détecte le support libre). On néglige également le temps de propagation sur le support.

a Remplissez un diagramme des temps, gradué en ST, décrivant le déroulement des différentes transmissions de trames, en adoptant la convention suivante :

A	*slot occupé par A*
X	*slot occupé par une collision*
	slot vide

et en supposant que les valeurs aléatoires générées par les machines *B*, *C* et *D* soient les suivantes :

	B	**C**	**D**
après 1 collision	0	1	1
après 2 collisions	2	1	1
après 3 collisions	4	5	1

b Calculez, sur la période allant de $t = 0$ à la fin de la transmission de la dernière trame, le taux d'utilisation du canal pour la transmission effective des 4 trames.

Solution **a** Le chronogramme est le suivant :

ST	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
	A	*A*	*A*	*A*	*A*	*A*	*X*	*B*	*B*	*B*	*B*	*B*	*B*	*X*		*X*		*D*	*D*	*D*	*D*	*D*	*D*	*C*	*C*	*C*	*C*	*C*	*C*

Commentaire : À la date 0, *A* démarre, le support est libre et sa trame dure 6 ST donc de 0 à 5 ST. À *t* = 2 ST, *B* et *C* veulent transmettre mais le support est occupé : elles attendent. À *t* = 5 ST, *D* veut transmettre, le support est occupé, donc elle attend.

À *t* = 6 ST, le support devient libre, toutes les stations en attente (*B*, *C* et *D*) tentent leur chance : il y a collision. *B*, *C* et *D* suspendent leur transmission et démarrent une attente aléatoire. Celle-ci sera nulle pour *B* et de 1 ST pour les deux autres. À *t* = 7 ST, *B* tente sa chance une nouvelle fois. Le support est libre, sa trame dure 6 ST, elle va de 7 à 12 ST.

À *t* = 8 ST, *C* et *D* veulent faire une nouvelle tentative. Le support étant occupé, elles attendent.

À *t* = 13 ST, le support devient libre. Toutes les stations en attente (*C* et *D*) tentent leur chance : il y a une nouvelle collision. *C* et *D* suspendent leur transmission et démarrent une deuxième attente aléatoire, valant 1 ST pour chacune, conformément au tableau précédent.

À *t* = 14 ST, il y a un silence, car les deux stations *C* et *D* attendent la fin du délai aléatoire et à *t* = 15 ST, elles tentent leur chance, une nouvelle fois ensemble ! Il y a à nouveau collision. Cette fois, le délai aléatoire est heureusement différent pour les deux stations qui vont donc réussir à transmettre : pour *D* à *t* = 17 ST et pour *C* à *t* = 23 ST puisque à sa troisième tentative (à *t* = 16 + 5 = 21 ST), le support est occupé par *D*.

b Le taux d'utilisation du canal est de 24/29 soit de 82 %.

EXERCICE 11 RISQUE DE COLLISIONS ET DÉLAI MOYEN D'ATTENTE

Énoncé

On suppose que l'algorithme de calcul du délai aléatoire après collision est celui de l'exercice précédent.

a Deux équipements *A* et *B* sur un réseau local Ethernet entrent en collision. Pour *A*, il s'agit d'une première collision, alors que pour *B*, il s'agit de la seconde. Quelle est la probabilité qu'il y ait une nouvelle collision entre *A* et *B* à leur prochaine tentative ?

b Même question avec une première collision pour *A* et la cinquième pour *B*.

c Calculez le temps moyen T_n d'attente cumulé pour l'accès au support d'un équipement qui a subi *n* collisions successives pour une trame donnée.

Solution **a** *A* n'a subi qu'une collision, donc le délai aléatoire qu'il a tiré au sort est 0 ou 1 fois l'intervalle ST. *B* en a subi deux successives, donc le délai qu'il a pu tirer au sort est uniformément réparti entre 0 ST, 1 ST, 2 ST et 3 ST.

Soit *p* la probabilité d'une nouvelle collision. Pour qu'un tel événement se produise, il faut que les deux équipements aient tiré au sort simultanément 0 ou simultanément 1. Notons *NA* (respectivement *NB*) la durée du délai pour *A* (respectivement *B*). Nous obtenons :

$$p = \text{Proba}\,[NA = 0]*\text{Proba}\,[NB = 0] + \text{Proba}\,[NA = 1]*\text{Proba}\,[NB = 1]$$

$$p = 1/2*1/4 + 1/2*1/4 = 1/4 = 0{,}25.$$

b Si B a déjà subi 5 collisions, le délai qu'il va tirer est réparti entre 0 ST et 31 ST.

$$p = \text{Proba}\ [NA = 0]*\text{Proba}\ [NB = 0] + \text{Proba}\ [NA = 1]*\text{Proba}\ [NB = 1]$$

$$p = 1/2*1/32 + 1/2*1/32 = 1/32.$$

c Le nombre de collisions déjà subies par un équipement permet de déterminer la taille de l'intervalle dans lequel il tire au sort son délai d'attente. Le temps moyen d'attente avant retransmission pour un essai donné est en effet égal à la moitié de l'intervalle de tirage, puisqu'il s'agit d'une loi uniforme. Le temps moyen cumulé pour n tentatives est donc la somme de chaque temps moyen, pour n allant de 1 à 16.

Soit ST la durée du Slot-Time.

Si $n = 0$, l'équipement n'a subi aucune collision et $T_0 = 0$.

Dans le cas où l'équipement a subi n collisions au total, avec n inférieur ou égal à 10, avant de réussir sa transmission, son délai d'attente se calcule comme suit :

Le délai d'attente a une valeur nulle avant la première transmission. Après la première collision, comme N vaut 0 ou 1, on a $D_1 = (0 + 1)/2*ST = ST/2$. Après la deuxième collision, N vaut 0, 1, 2 ou 3, donc on a $D_2 = (0 + 1 + 2 + 3)/4*ST = 3\ ST/2$. Après la troisième collision, N vaut 0, 1, 2, 3, 4, 5, 6 ou 7. On obtient : $D_3 = (0 + 1 + 2 + 3 + 4 + 5 + 6 + 7)/8*ST = 7\ ST/2$ et ainsi de suite jusqu'à n. Donc :

$$Tn = D0 + D1 + D2 + \ldots + Dn = 0 + ST/2 + 3\ ST/2 + \ldots$$
$$+ (2^{n-}1)\ ST/2 = (2^{n-}(n + 1)/2)*ST.$$

Si l'équipement a subi n collisions au total, avec n compris entre 11 et 15 (bornes incluses), le calcul est légèrement différent du précédent puisque : $D10 = D11 = D12 = D13 = D14 = D15$.

On trouve alors :

$$Tn = T10 + (n - 10)*D10.$$

Exercice 12 Latence d'un anneau à jeton

Énoncé

Un réseau local en anneau comprend 10 stations uniformément réparties sur l'anneau. La vitesse de propagation des signaux est de 200 m/µs. Les trames ont une longueur totale de 256 bits. Calculez le nombre de bits en transit sur l'anneau pour les configurations suivantes :

a Pour une longueur de 10 km et un débit binaire de 5 Mbit/s.

b Pour une longueur de 1 km et un débit binaire de 500 Mbit/s.

c Comparez les deux anneaux du point de vue du nombre de trames en transit, du débit utile et du rendement, si la station émettrice attend le retour de sa propre trame pour réinjecter un jeton sur l'anneau.

Solution

a Le débit est 5 Mbit/s, donc 1 bit dure $1/(5*10^6) = 0{,}2$ µs. La vitesse de propagation étant de 200 m/µs, 1 bit équivaut à 200*0,2 = 40 m de câble. Si la longueur de l'anneau est de 10 km, la latence vaut : 10 000/40 = 250 bits.

b Le débit est 500 Mbit/s, donc 1 bit dure $1/(500*10^6) = 0{,}002$ µs. La vitesse de propagation étant de 200 m/µs, 1 bit équivaut à 200*0,002 = 0,40 m de câble. Si la longueur de l'anneau est de 1 km, la latence est alors de 1 000/0,40 = 2 500 bits.



c Le nombre de trames en transit dans le cas a est presque de 1, puisque le temps de propagation sur l'anneau est de 10 000/200 = 50 µs et que le temps de transmission d'une trame vaut : $256/(5*10^6) = 51{,}2$ µs. Un équipement émet le 251e bit quand il reçoit le 1er. Il émet le 252e bit quand il reçoit le 2e… Pour transmettre le jeton, il doit attendre d'avoir reçu son 256e bit. Il attend donc 50 µs. Le débit utile est de $(5*10^6*51{,}2)/(51{,}2 + 50) = 2{,}52$ Mbit/s. Le rendement vaut : 2,52/5 = 50 %.

Dans le cas b, le nombre de trames en transit est presque de 10, car le temps de propagation sur l'anneau est de 1 000/200 = 5 µs et le temps de transmission d'une trame est de $256/(500*10^6) = 0{,}512$ µs. Dans la latence de l'anneau (2 500 bits) il pourrait y avoir 2 500/256 = 9,76 trames. Or, pour pouvoir réinjecter le jeton, l'équipement doit attendre d'avoir reçu la fin de sa propre trame. Il attend donc 5 µs et le débit utile est de $(500*10^6*0{,}512)/(0{,}512 + 5) = 46{,}4$ Mbit/s. Le rendement est nettement plus faible : 46,4/500 = 9 %.

Remarque

Cette perte d'efficacité est caractéristique du mécanisme à base de jeton quand le débit augmente. Elle explique pourquoi les réseaux de type Token Ring ont moins bien évolué que les réseaux Ethernet vers les Gbit/s. Pour conserver une bonne efficacité, il aurait fallu changer le mode de gestion du jeton, donc changer les cartes réseau de toutes les machines, ce qui représente un investissement considérable. Une variante de l'anneau à jeton (FDDI, *Fiber Distributed Data Interface*) sur fibre optique, fonctionnant à 100 Mbit/s, utilise des jetons temporisés et autorise la présence de plusieurs jetons dans l'anneau.

EXERCICE 13 SIMULATION DE TRAFIC SUR UN ANNEAU À JETON

Énoncé

Soit un anneau à jeton constitué de 4 stations *A*, *B*, *C* et *D*. À un instant donné, *A* émet une trame avec la priorité 2. *C* veut émettre une trame avec la priorité 5 et *D* veut émettre avec la priorité 7. Sachant que chaque station s'occupe de retirer de l'anneau la trame qu'elle a émise et qu'elle doit ensuite remplacer celle-ci par un jeton libre, indiquez comment les stations opèrent pour répondre aux besoins du trafic.

Solution

A possède le jeton et transmet sa trame de priorité 2 qui arrive à la station voisine *B* qui n'a pas de trafic. *B* se contente donc de la répéter vers *C*. La station *C* a une trame à émettre et celle-ci est de priorité 5. Dans la trame de *A* qu'elle reçoit de *B* et répète vers *D*, *C* positionne le champ de réservation de priorité qui était vide à la valeur 5. La station *D* a une trame à émettre de priorité 7. Dans la trame de *A* qu'elle reçoit de *C* et répète vers *A*, elle remplace le contenu du champ de réservation de priorité par la valeur 7 (et elle mémorise la valeur 5 qu'elle vient d'écraser). Lorsque *A* a fini de recevoir sa propre trame, elle envoie donc un jeton libre vers *B* avec la priorité 7, que *B* laisse passer. *C* reçoit le jeton mais celui-ci est de priorité plus élevée que celle demandée. Elle laisse donc passer le jeton à destination de *D*. Celle-ci reçoit le jeton qui correspond à la priorité réclamée. *D* prend le jeton et transmet sa trame de priorité 7 qui fait le tour de l'anneau.

Au passage de cette trame, la station *C* qui a toujours une trame en attente, laisse encore passer son tour. Lorsque *D* a fini de recevoir sa propre trame, elle envoie un jeton libre vers *A* avec la priorité 7 (cela permet à toutes les autres stations qui auraient une trame de priorité 7 d'écouler leur trafic. Le jeton libre revient à *D* puisque ici il n'y a pas d'autre trafic de priorité 7). *D* envoie enfin un jeton libre avec la priorité 5 que *C* souhaitait. *C* transmet sa

trame de priorité 5 qui fait tout le tour de l'anneau puis elle réinjecte enfin un jeton libre de même priorité (5) avant d'envoyer un jeton à la priorité précédente (2).

Remarque

Le niveau de priorité 5 est *a priori* réservé aux stations qui ont des fonctions particulières (les ponts, par exemple). La priorité maximale 7 est utilisée pour les trames de supervision de l'anneau. En absence de demande, le jeton libre circule avec la priorité 0, donc toutes les stations peuvent émettre, même avec la priorité la plus faible. L'exercice donne un aperçu de la complexité de gestion des priorités dans un anneau à jeton.

EXERCICE 14 ETHERNET COMMUTÉ

Énoncé

Soit un réseau Ethernet commuté constitué de 45 équipements et d'un serveur connectés à un commutateur 100 Base T.

a Quelle est la topologie physique de ce réseau ? Quel est le débit du réseau et quel support de transmission est utilisé ?

b À l'aide de cet exemple, montrez les principales différences de fonctionnement entre un concentrateur et un commutateur.

c Si 5 équipements transmettent des données simultanément vers le serveur, quel débit théorique peut espérer chacun d'entre eux ?

Solution

a La topologie physique est en étoile, le débit de 100 Mbit/s sur paires métalliques.

b Avec un concentrateur, lorsqu'un équipement émet vers un autre, tous les équipements du réseau reçoivent l'information. Le débit de 100 Mbit/s est partagé entre les utilisateurs et les transferts de données se font à l'alternat. Un concentrateur est un équipement très bon marché.

Avec un commutateur, si un équipement émet vers un autre, seul le destinataire reçoit l'information. Chaque utilisateur emploie un débit de 100 Mbit/s et les transferts de données sont bidirectionnels simultanés. Un commutateur est plus onéreux mais le rapport prix/performances vaut le supplément.

c Si le commutateur a une capacité suffisante, chaque équipement, directement relié au commutateur, peut disposer d'un débit théorique dédié de 100 Mbit/s dans les deux sens de transmission. Puisque les 5 équipements communiquent avec le même serveur, le lien entre le serveur et le commutateur est en fait partagé entre les 5 communications : un débit maximal de 20 Mbit/s est offert à chaque dialogue.

EXERCICE 15 GIGABIT ETHERNET

Énoncé

Pour les transmissions de type Ethernet 1 Gbit/s, la trame doit avoir une longueur minimale de 512 octets.

a Quel est le temps d'émission d'une trame de longueur minimale ?

b Peut-on en déduire la période de vulnérabilité dans un tel réseau ?

Solution

a Le temps d'émission d'une trame de 512 octets est $512*8/10^9$ soit environ 4 µs.

b On peut en déduire que la période de vulnérabilité est au plus égale à 4 µs.

> **Remarque**
>
> On comprend ici pourquoi il a fallu augmenter la taille de la trame : la période de vulnérabilité aurait été trop courte.

EXERCICE 16 RÉSEAUX LOCAUX VIRTUELS

Énoncé

On considère un commutateur avec 8 ports numérotés *P1*, *P2*, …, *P8*. 6 équipements notés *M1*, *M2*, …, *M6* sont connectés à ce commutateur. *M1* est connecté sur le port *P1*, *M2* sur le port *P2*…

a L'équipement *M1* envoie une trame Ethernet avec l'adresse de diffusion. Qui reçoit cette trame ? Si maintenant l'équipement *M1* envoie une trame Ethernet à *M3*, qui la reçoit ?

b On met en place des réseaux locaux virtuels par port sur le commutateur : le VLAN *A* contient les équipements *M1*, *M3* et *M5*, le VLAN *B* les équipements *M2*, *M4* et *M6*. Donnez l'affectation des ports aux différents VLAN dans le commutateur.

c Que se passe-t-il pour le scénario de la question a ?

d On ajoute un second commutateur avec la même configuration et 6 nouveaux équipements *M11*, *M12*, *M13*, *M14*, *M15* et *M16*. Les équipements de numéro impair appartiennent au VLAN *A*, ceux de numéro pair au VLAN *B*. Proposez une solution pour relier les deux commutateurs. Donnez la nouvelle affectation des ports aux différents VLAN dans le premier commutateur.

e Peut-on n'utiliser qu'un seul lien entre les deux commutateurs ?

Solution

a Lorsque *M1* envoie une trame Ethernet avec l'adresse de diffusion, le commutateur la répète sur l'ensemble de ses ports : tous les équipements, de *M1* à *M6*, la reçoivent. Quand *M1* envoie ensuite une trame à *M3*, le commutateur la reçoit sur le port *P1* et la transmet sur le port *P3* : seul *M3* la reçoit.

b Le commutateur associe *P1*, *P3* et *P5* au VLAN *A* et *P2*, *P4* et *P6* au VLAN *B*.

c Le commutateur diffuse au sein du VLAN *A* la trame de *M1* arrivant par le port *P1*. Les équipements de numéros pairs ne la reçoivent pas : le commutateur isole les équipements des deux VLAN, le trafic de l'un ne passe pas sur l'autre. Le traitement de la trame envoyée par *M1* à *M3* est inchangé, puisque *M1* et *M3* sont à l'intérieur du même VLAN *A*.

d Le second commutateur peut avoir une table semblable à celle du premier : *P1*, *P3* et *P5* appartiennent au VLAN *A* et *P2*, *P4* et *P6* au VLAN *B*. Il reste à relier les deux commutateurs. Pour cela, on peut relier les ports 7 de chaque commutateur et affecter ce port au VLAN *A*. De même, on peut relier les ports 8 de chacun d'eux et l'affecter au VLAN *B*.

La table devient alors : *P1*, *P3*, *P5* et *P7* au VLAN *A* et *P2*, *P4*, *P6* et *P8* au VLAN *B*.

e Si on met deux liens (en reliant les deux ports *P7* entre eux et les deux ports *P8* entre eux), on se retrouve avec une boucle dans le réseau et il faudra gérer l'algorithme de l'arbre couvrant. Il n'y a pas de boucle si on ne met qu'un seul lien entre les deux commutateurs (entre les ports *P7*, par exemple), mais les deux VLAN doivent alors partager ce lien : le

port *P7* appartient aux deux VLAN. Dans ce cas, il faut que les VLAN soient étiquetés pour que les commutateurs sachent comment traiter les trames. Il faut également que les cartes réseau des équipements supportent le standard 802.1Q qui permet l'étiquetage des VLAN.

Remarque

Cet exercice montre l'intérêt d'utiliser le protocole 802.1Q : en permettant la création de plusieurs arbres couvrants, on peut relier les deux commutateurs par plusieurs *trunks* et donc améliorer la tolérance aux pannes du réseau.

EXERCICE 17 INTERCONNEXION

Énoncé

Deux entreprises *A* et *B*, installées dans le même immeuble de bureaux, sont équipées l'une d'un réseau local de type Ethernet, l'autre d'un réseau local de type Token Ring.

a Proposez une solution d'interconnexion pour que chaque station de l'entreprise *A* puisse dialoguer avec toutes les stations de l'entreprise *B*.

b À quoi pourrait-on attribuer le goulet d'étranglement et la dégradation éventuelle des débits utiles d'une station du réseau *A* ?

Solution

a Les deux réseaux sont proches l'un de l'autre mais ils utilisent des couches MAC différentes. Il est donc impossible de les relier par des répéteurs. Un pont au minimum est indispensable, mais un routeur pourra tenir compte de la taille des trames différentes : en constatant qu'une trame de taille maximale de Token Ring ne peut pas entrer telle quelle dans un réseau Ethernet, il fabrique autant de trames que nécessaire pour ne pas perdre d'informations entre les deux réseaux locaux, ce qu'un simple pont ne sait pas faire. La réponse à la question suivante explique pourquoi.

b Si les débits sont différents (10 Mbit/s sur Ethernet et 16 Mbit/s sur Token Ring par exemple), une longue rafale de trafic dans le sens *B* vers *A* peut entraîner des pertes de trames, parce que le pont ne peut plus les mémoriser avant de les envoyer vers le réseau le plus lent. En outre, s'il y a plusieurs trames de longueur maximale (4 500 octets) de *B* vers *A*, celles-ci doivent être fragmentées dans le réseau *A* : il faut donc un équipement capable de fragmenter le champ de données trop long, puis d'encapsuler chaque fragment dans une trame au format adéquat. Seuls un routeur ou un commutateur-routeur peuvent assurer ces fonctions.

D'autre part, le service rendu par les deux couches MAC est différent. Une station du réseau *B* ne reçoit pas l'accusé de réception (champ FS de la trame) qu'elle attend. En effet, le pont qui se trouve dans le réseau *A* ne renvoie pas la trame, comme le fait une station d'un anneau à jeton. Dans le cas où le pont a rempli le champ FS de lui-même, la machine du réseau *B* croit que son destinataire a bien reçu la trame, ce qui n'est peut-être pas vrai.

Dans le cas où Ethernet est le réseau local le plus rapide, il n'y a pas de risque de trouver des champs de données trop longs mais la mémoire du pont peut être insuffisante pour éponger une rafale prolongée de trafic vers le réseau Token Ring.



EXERCICE 18 RÔLE DES PONTS

Énoncé

a Soit le réseau suivant (voir figure 5.19) constitué de trois sous-réseaux Ethernet. Le protocole de niveau réseau utilisé est IP. Les machines *D* et *H* sont des ponts. Décrivez l'envoi de la trame de *F* à *C* (une station qui vient de se connecter au réseau). Si *K* est un analyseur de trafic, peut-il enregistrer la trame ?

Figure 5.19
Exemple de réseau.

b Même question avec la trame réponse de *C* à *F*.

c On étend le réseau précédent (voir figure 5.20). Les équipements d'interconnexion *D*, *E*, *I* et *H* peuvent-ils être des répéteurs ? Des ponts ?

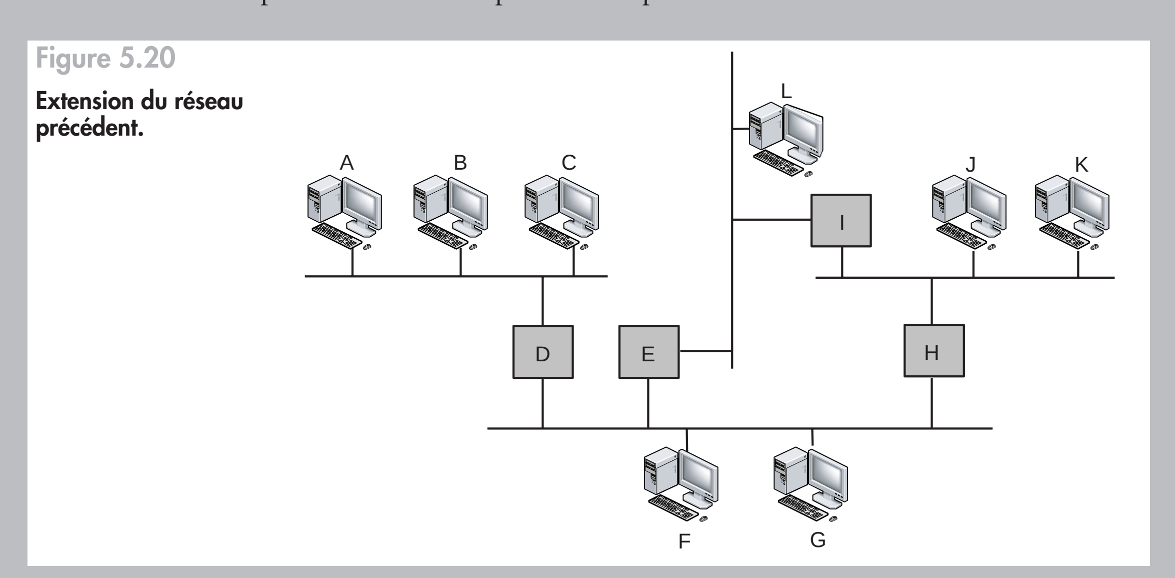


Figure 5.20
Extension du réseau précédent.

Solution

a Si *D* et *H* sont des ponts, ils reçoivent l'un et l'autre la trame émise par *F*. Les deux ponts, ne sachant pas où se trouve le destinataire, laissent passer la trame, laquelle circule sur les deux autres sous-réseaux. *C* reçoit donc la trame. *K*, qui est dans le troisième sous-réseau, la voit passer de même et peut l'enregistrer.

b Lorsque *C* répond à *F*, la trame diffusée parvient au pont *D* qui sait qu'il doit la laisser passer puisque *F* est accessible de l'autre côté. Par contre, le pont *H* sait que *C* et *F* sont situés

du même côté. Il filtre donc la trame qui ne transite pas sur le troisième sous-réseau : l'analyseur de protocole *K* ne voit pas la trame réponse.

c Si *D*, *E*, *I* et *H* sont des répéteurs, le réseau ne peut pas fonctionner car il n'a plus sa structure de bus ramifié. Si *D*, *E*, *I* et *H* sont des ponts, le réseau ne peut fonctionner que si l'un des ponts est inactif (ce qui a pour effet de « couper » la boucle). Les ponts constituent un ensemble collaboratif, ils discutent entre eux et décident celui qui sera inactif (c'est le rôle de STP, l'algorithme de l'arbre couvrant).

> **Remarque**
>
> L'explication de la question a suppose que les ponts fassent de l'autoapprentissage. Si les ponts sont configurés à l'avance avec les adresses MAC des stations qui sont de chaque côté, seul le pont *D* laisse passer la trame qui va de *F* à *C* ainsi que sa réponse de *C* à *F*. L'analyseur *K* ne voit rien de ce trafic. Cet exemple montre l'intérêt des ponts pour segmenter des réseaux locaux.

Exercice 19 Algorithme de l'arbre couvrant

Énoncé Considérons un réseau d'entreprise composé de segments Ethernet interconnectés par des commutateurs, comme le montre la figure 5.21.

Figure 5.21
Interconnexion des segments Ethernet par des commutateurs.

Eth 1
Eth 4
Eth 3
Eth 2

a Considérons les réseaux Eth1 et Eth2. Combien de chemins différents les données peuvent-elles emprunter pour aller d'un réseau à l'autre ?

b Comment sera assurée la redondance en cas de panne d'un commutateur ?

c Que peut-il arriver si un des commutateurs n'arrive plus à recevoir d'informations sur un port mais qu'il puisse toujours en émettre ?

Les BPDU de configuration sont constitués du triplet : < *ID pont racine.coût à la racine. ID pont émetteur* >. Un commutateur d'identifiant ID = 27 reçoit sur ses différents ports les BPDU : *port 1* : < 35.2.48 > ; *port 2* : < 15.3.17 > ; *port 3* : < 15.3.19 > ; *port 4* : < 15.3.22 >.

d Quel est le meilleur BPDU de configuration que ce commutateur peut fabriquer ?

e Quelle valeur maximale faut-il donner à l'ID du commutateur pour qu'il puisse devenir racine ?

f Quel est le port racine ? Quels ports font partie de l'arbre couvrant ?



Solution

a Avant le transfert des données, les commutateurs exécutent l'algorithme STP pour éviter les boucles dans le réseau. Une fois que les commutateurs l'ont exécuté, il n'y a qu'un seul chemin possible pour les données entre Eth1 et Eth2.

b Certains commutateurs ne vont plus recevoir de BPDU de configuration et cela provoquera une nouvelle exécution de l'algorithme de l'arbre couvrant pour construire une nouvelle arborescence.

c Il ne va plus recevoir de BPDU de configuration sur le port en panne. Il commence donc une nouvelle exécution de l'algorithme de l'arbre couvrant de manière intempestive. Comme il ne reçoit plus les informations, il peut déclencher une tempête de diffusion (*broadcast storm*). Il faut le déconnecter !

d Le meilleur BPDU de configuration qu'il puisse produire est : `< 15.4.27 >`.

e Il faut que l'ID ait au maximum la valeur 14.

f Le port racine est le port 2. Les ports 1 et 2 font partie de l'arbre couvrant.

EXERCICE 20 UTILISATION DE VRRP POUR ÉQUILIBRER LE ROUTAGE DANS UN RÉSEAU D'ENTREPRISE

Énoncé

Une entreprise utilise deux serveurs pour les machines de son réseau constitué de quatre VLAN. Jusqu'à présent, un simple commutateur raccorde les différents VLAN aux deux serveurs, comme le montre la figure 5.22.

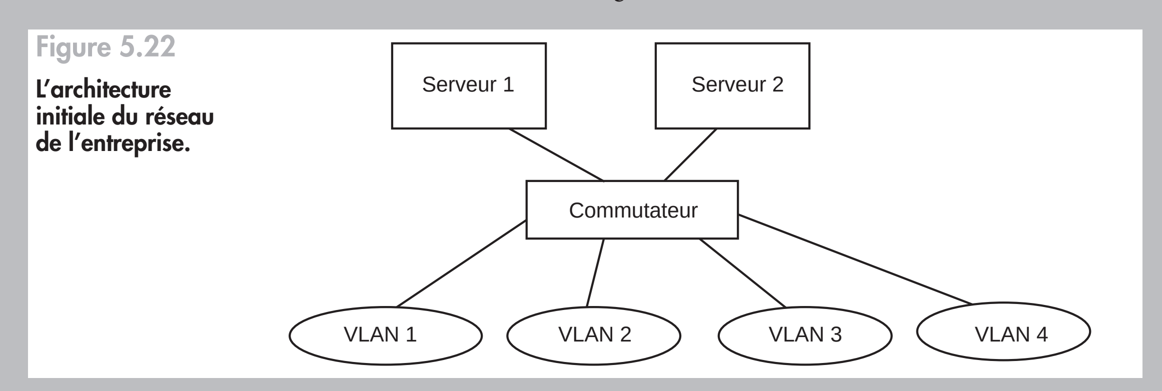


Figure 5.22
L'architecture initiale du réseau de l'entreprise.

a L'accès aux serveurs devient une ressource critique. L'entreprise veut équilibrer le routage entre ses quatre VLAN, le serveur 1 étant principalement utilisé par les VLAN 1 et 2, le serveur 2 par les VLAN 3 et 4. Proposez une solution qui puisse équilibrer le routage et permette au réseau de continuer à fonctionner si l'un des équipements d'interconnexion tombait en panne.

☞

Énoncé

b Le réseau évolue car l'entreprise grandit. Elle acquiert un deuxième commutateur pour constituer un *cœur de réseau*, comme le montre la figure 5.23. L'administrateur constate alors une saturation des liens de son réseau. Pouvez-vous expliquer pourquoi ?

Figure 5.23

Relations enre le cœur de réseau et les serveurs de l'entreprise.

c Proposez une solution pour y remédier tout en préservant la redondance de l'architecture.

Solution

a Pour introduire une redondance dans l'accès aux VLAN, il faut doubler le dispositif qui assure l'acheminement dans le réseau : le plus simple est d'installer deux commutateurs-routeurs utilisant le protocole VRRP pour raccorder chaque serveur par deux liens différents (en *double attachement* : un lien est actif pendant que l'autre est désactivé). Les deux commutateurs-routeurs constituent le cœur du réseau de la figure 5.23.

b Avec l'adjonction du nouveau commutateur, on a créé la boucle : Serv 1 – commutateur 1 – Serv 2 – commutateur 2 – Serv 1. Cela explique pourquoi les liens sont saturés.

c Pour supprimer la boucle, la première des choses à faire est d'activer l'arbre couvrant (STP). Pour privilégier un chemin en fonctionnement normal sans interdire la redondance, il faut créer des chemins différents, donc utiliser des arbres couvrants multiples, en construisant un STP par VLAN selon le standard 802.1Q.

La solution précédente est à compléter car, en cas de défaillance d'un équipement, le chemin entre un VLAN donné et le serveur auquel il fait le plus souvent appel n'est pas forcément simple. On a donc intérêt à ajouter un lien entre commutateur 1 et commutateur 2 pour simplifier le trajet dans le réseau.



Le protocole IP (*Internet Protocol*)

IP transfère les données à travers une interconnexion de réseaux. Il est utilisé par les protocoles de la couche de transport, TCP et UDP. Il cherche un chemin pour transférer les données (*datagrammes*) d'un équipement émetteur, identifié par son adresse IP, à un équipement destinataire, identifié lui aussi par son adresse IP. Dans une machine quelconque, le module IP ne fournit aucune garantie d'un acheminement correct des données et ne gère aucun dialogue avec le module IP d'une autre machine. Chaque datagramme est géré indépendamment des autres. Cela signifie que ceux-ci peuvent être mélangés, dupliqués, perdus ou altérés ! Pour comprendre le fonctionnement du protocole IP, nous allons d'abord voir les adresses IP elles-mêmes ainsi que leur correspondance avec les adresses physiques, le traitement effectué par un module IP et le format du datagramme IP.

1 Les adresses IP (*Internet Protocol*)

L'adressage utilisé dans Internet est un adressage logique. Chaque équipement possède un nom symbolique auquel on fait correspondre l'adresse logique appelée *adresse IP*. Celle-ci se décompose en deux parties : l'*identificateur du réseau*, où se trouve l'équipement, et l'*identificateur de la machine* elle-même (qui a une signification locale à ce réseau). L'ensemble tient sur 32 bits soit 4 octets. L'adresse IP est le plus souvent écrite en *notation décimale pointée* : les octets sont séparés par des points, et chaque octet est représenté par un nombre décimal compris entre `0` et `255`.

Exemple

Adresse IP = `11000001 00011011 00101101 00100001` en binaire soit `193.27.45.33` en notation décimale pointée.

1.1 Les classes d'adresses

Plusieurs classes d'adresses sont définies : un réseau ayant beaucoup de machines dispose d'une adresse avec un champ *identificateur de réseau* court et un champ *identificateur de machine* long. En revanche, dans un petit réseau local, l'identificateur de machine sera codé sur peu d'éléments binaires. La classe d'adresse et l'identificateur de réseau sont attribués par un organisme central, l'ICANN (*Internet Corporation for Assigned Names and Numbers*), qui gère le plan d'adressage à l'échelle mondiale et garantit l'unicité des numéros de réseau. L'administrateur local du réseau attribue ensuite les numéros de machine aux différents équipements de son réseau, selon le plan d'adressage qu'il a conçu.

L'identificateur de réseau est codé sur 7, 14 ou 21 bits selon la classe d'adresse. Les adresses de classe *A* affectent 7 bits à l'identité de réseau et 24 bits à l'identité de machine ; les adresses de classe *B* affectent 14 bits à l'identité de réseau et 16 bits à l'identité de machine. Enfin, les adresses de classe *C* allouent 21 bits à l'identité de réseau et 8 bits à l'identité de machine. Les adresses de classe *D* sont réservées pour mettre en œuvre le mécanisme de diffusion de groupe. La classe d'une adresse IP est déterminée à partir des bits de poids fort, comme le montre la figure 6.1.

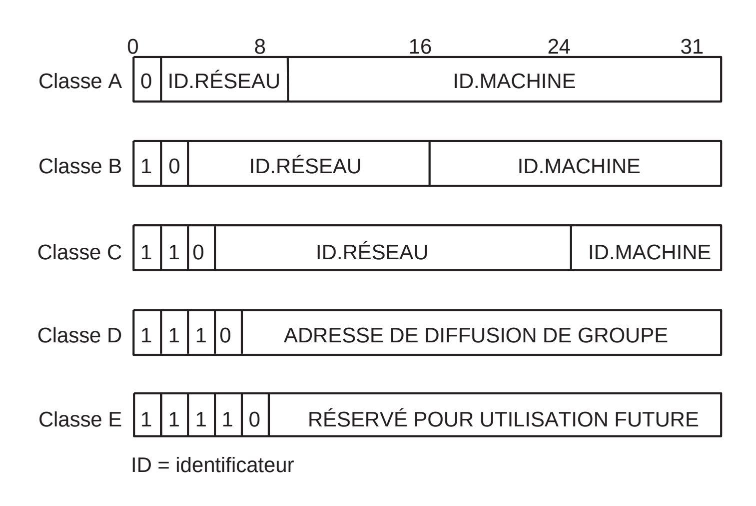


Figure 6.1

Différents formats d'adresse IP.

Les très grands réseaux ont des adresses de classe *A*, dont le premier bit du premier octet est à 0 tandis que les 7 autres bits servent à identifier 126 réseaux différents. Chaque réseau de classe *A* possède 24 bits d'identifiant de machine, ce qui permet d'adresser $2^{24} - 2$, soit 16 777 214 machines (les deux identifiants 0 et 16777215 sont, par convention, réservés à un autre usage).



Les réseaux de taille moyenne ont des adresses de classe *B*. Elles commencent en binaire par `10` et affectent 14 bits à l'identifiant de réseau ; il reste 16 bits pour identifier les machines, soit au maximum 65 534 (pour la même raison que précédemment, les identifiants 0 et 65535 ne sont pas attribués à une machine).

Enfin, pour les petits réseaux, les adresses de classe *C* commencent en binaire par `110` et allouent 21 bits à l'identifiant de réseau et 8 bits à l'identifiant de machine. On peut ainsi adresser jusqu'à 254 machines (les identifiants 0 et 255 ne sont pas utilisés).

Les adresses de classe *D*, commençant en binaire par `1110`, sont réservées à la mise en œuvre d'un mécanisme de diffusion de groupe (*multicast*). Dans une communication multicast, un utilisateur émet un message dont l'adresse de destination est celle du groupe. Le message est acheminé en un seul exemplaire le plus loin possible, jusqu'à ce qu'il soit indispensable de l'éclater en autant de messages individuels que le groupe possède de membres. La plupart des adresses multicast allouées le sont à des groupes d'utilisateurs concernés par une même application (la radio sur Internet par exemple. On comprend dans ce cas l'intérêt d'envoyer en multicast au lieu d'envoyer massivement autant de messages qu'il y a de récepteurs). Dans une adresse multicast, les 28 bits restants n'ont pas de structure particulière.

Exemple

Adresse IP = `11000001 00011011 00101101 00100001` soit `193.27.45.33`. Il s'agit d'une adresse de classe *C* puisqu'elle commence en binaire par `110`. Le découpage est alors le suivant : `110` (3 bits pour la classe *C*) `00001 00011011 00101101` (21 bits d'identifiant de réseau) et `00100001` (8 bits identifiant la machine dans le réseau).

1.2 Adresses particulières

Lorsqu'une machine ne possède pas d'adresse IP et qu'elle doit envoyer un (premier) message pour en obtenir une, elle remplit le champ Adresse du message par « plein 0 » ou `0.0.0.0` en notation décimale pointée. À l'opposé, remplir le champ Adresse par « plein 1 » permet de désigner l'ensemble des machines au sein du réseau dans lequel se trouve la machine.

Les réseaux eux-mêmes possèdent chacun une adresse : celle-ci est obtenue en remplaçant le champ Identifiant de machine par « plein 0 », conformément à la classe.

Exemple

La machine `37.194.192.21` appartient au réseau `37.0.0.0` (classe *A*).
La machine `137.194.192.21` appartient au réseau `137.194.0.0` (classe *B*).
La machine `197.194.192.21` appartient au réseau `197.194.192.0` (classe *C*).

Une *adresse de diffusion (broadcast address)*, désigne l'ensemble des machines d'un réseau distant. Elle est constituée en remplaçant le champ Identifiant de machine par « plein 1 ».

Exemple

L'adresse de diffusion du réseau `37.0.0.0` est `37.255.255.255` (classe *A*).
L'adresse de diffusion du réseau `137.194.0.0` est `37.194.255.255` (classe *B*).
L'adresse de diffusion du réseau `197.194.192.0` est `197.194.192.255` (classe *C*).

Certains identifiants de réseau n'existent pas, en particulier les réseaux de classe *A* : `0` et `127. 0` est réservé à l'usage décrit ci-avant et `127` sert aux tests locaux. Par exemple, l'adresse `127.0.0.1` est *a priori* affectée à chaque carte réseau. Tout message envoyé à cette adresse est directement retourné à son expéditeur, sans aucune émission sur le réseau : cela permet de vérifier que la pile TCP/IP fonctionne correctement. Notons que cette adresse, appelée *adresse de boucle locale (loopback address)*, n'a aucun rapport avec la notion de boucle locale utilisée pour la desserte des usagers du réseau téléphonique.

1.3 Notions de sous-réseaux et de masque

La hiérarchie à deux niveaux (réseau et machine) de l'adressage IP s'est rapidement révélée insuffisante à cause de la diversité des architectures des réseaux connectés. La notion de sous-réseau (ou *subnet*), introduite en 1984, a conservé le format de l'adresse IP sur 32 bits. Dans un réseau subdivisé en plusieurs sous-réseaux, on exploite autrement le champ Identifiant de machine de l'adresse IP. Celui-ci se décompose désormais en un identifiant de sous-réseau et un identifiant de machine. Remarquons que ce découpage n'est connu qu'à l'intérieur du réseau lui-même. En d'autres termes, une adresse IP, vue de l'extérieur, reste une adresse sur 32 bits avec ses deux champs. On ne peut donc pas savoir si le réseau est constitué d'un seul réseau ou subdivisé en plusieurs sous-réseaux.

L'administrateur local choisit le nombre de bits à consacrer à l'identifiant de sous-réseau grâce au *masque de sous-réseau* (ou *subnet mask*). Celui-ci, également codé sur 32 bits, définit le découpage de l'identifiant machine en deux champs (Sous-réseau et Machine). Dans un réseau subdivisé, chaque machine connaît son adresse IP et le masque utilisé, ce qui lui permet de savoir dans quel sous-réseau elle se trouve. Il suffit de faire un ET logique entre son adresse IP et le masque :

Exemple

Adresse IP : `193. 27. 45. 33`	=	`11000001 00011011 00101101 00100001`
Masque : `255.255.255.224`	=	`11111111 11111111 11111111 11100000`
Comparaison sous masque :		`11000001 00011011 00101101 00100001`
ET :		`11111111 11111111 11111111 11100000`
		`11000001 00011011 00101101 00100000`

Le résultat `193.27.45.32` est l'adresse du sous-réseau auquel appartient la machine `193.27.45.33`.

Lorsqu'un équipement d'un (sous-)réseau veut émettre un message à un autre, il compare sa propre adresse bit à bit avec celle du destinataire, en utilisant le masque de sous-réseau. S'il y a égalité sur toute la partie identifiée par les 1 du masque, les deux équipements se trouvent dans le même (sous-)réseau et le message peut donc être transmis directement. Sinon, il est envoyé au routeur, la machine qui assure l'acheminement du message vers l'extérieur du (sous-)réseau.

Exemple

Soit une adresse IP de réseau de classe *C* `193.27.45.0`. Le masque de sous-réseau vaut : `255.255.255.224`, soit en binaire : `11111111 11111111 11111111 11100000`.

Nous voyons donc que, dans l'octet réservé au champ Identifiant de machine, trois bits sont utilisés pour identifier des sous-réseaux interconnectés par un routeur. Sur le sous-réseau 1, l'adresse du sous-réseau est `193.27.45.32`. Si l'administrateur décide d'affecter l'identifiant 1 au routeur dans tous les sous-réseaux, l'adresse du routeur dans le sous-réseau 1 est : `193.27.45.33` ; l'adresse de diffusion dans le sous-réseau valant `193.27.45.63`, il reste donc 29 adresses disponibles sur les 32 possibles pour les stations du sous-réseau 1. De même, dans le sous-réseau 2, l'adresse de sous-réseau étant `193.27.45.64`, celle du routeur vaut `193.27.45.65` et l'adresse `193.27.45.95` est celle de diffusion dans le sous-réseau. Il reste également 29 adresses disponibles sur les 32 possibles pour les stations du sous-réseau 2.

Remarque

Dans un réseau plat (sans sous-réseau), on peut mettre 254 stations sur un réseau de classe *C*. Avec six sous-réseaux physiques comme dans cet exemple (on a exclu de fait les numéros `0` et `7`), on ne peut en mettre que 174, mais on dispose d'une identification plus fine et d'une possibilité de diffusion limitée à chaque sous-réseau.



1.4 Pénurie d'adresses

Le formidable succès d'Internet a mené à l'épuisement des adresses de classes *A* et *B* et à l'explosion des tables de routage des routeurs situés dans les réseaux de transit. Si beaucoup d'organisations possèdent plus de 254 ordinateurs, peu en possèdent plusieurs milliers (or, une adresse de classe *B* permet d'identifier jusqu'à 65 534 machines). Ce manque de souplesse entre les classes explique les différentes solutions mises en œuvre dès les années 1990 : l'utilisation d'adresses privées, d'adresses sans classe et la distribution dynamique des adresses.

Les adresses privées

Plusieurs plages d'adresses IP ont été réservées dans chaque classe d'adresses et sont d'utilisation libre. Elles sont appelées *adresses IP privées* et sont décrites dans la RFC 1918 (voir tableau 6.1).

Tableau 6.1

Les adresses IP privées en classes A, B et C

Classe	Adresses privées	Nombre maximal de machines
A	**10.x.y.z**, où 0 ≤ x ≤ 255 0 ≤ y ≤ 255 et 0 ≤ z ≤ 255	(256*256*256) – 2 = 16 777 214
B	**172.x.y.z**, où 16 ≤ x ≤ 31 0 ≤ y ≤ 255 et 0 ≤ z ≤ 255	(15*256*256) – 2 = 1 048 574
C	**192.168.x.y**, où 0 ≤ x ≤ 255 et 0 ≤ y ≤ 255	(256*256) – 2 = 65 534

Ces adresses ne peuvent être attribuées par l'ICANN à une organisation. Ainsi, des réseaux différents peuvent utiliser les mêmes adresses IP privées, pourvu qu'ils restent isolés les uns des autres.

Pour relier à Internet les machines d'un réseau utilisant des adresses privées, on met en place une traduction, gérée par le routeur, entre adresses IP privées (internes au réseau de l'organisation et inaccessibles de l'extérieur) et adresses IP publiques (visibles de l'extérieur, c'est-à-dire accessibles par Internet). Une adresse IP publique est unique ; elle est dite *routable*, car elle seule autorise l'accès à Internet. La correspondance entre les deux types d'adresses est assurée par le NAT (*Network Address Translation*), un mécanisme de conversion d'adresse décrit par la RFC 3022. De plus, les adresses IP privées garantissent une meilleure sécurité d'accès aux réseaux d'organisation, dans la mesure où les adresses réelles utilisées par les machines du réseau ne sont pas connues de l'extérieur.

Le NAT statique crée une bijection entre adresses IP publiques et adresses IP privées internes au réseau. Le routeur associe une adresse IP privée (par exemple `192.168.0.1`) à une adresse IP publique routable sur Internet. Il fait la traduction, dans un sens comme dans l'autre, en modifiant l'adresse dans le paquet IP. Le NAT statique connecte des machines du réseau interne à Internet de manière transparente, mais ne résout pas le problème de la pénurie d'adresses, dans la mesure où *n* adresses IP routables sont nécessaires pour connecter *n* machines du réseau interne.

Le NAT dynamique partage une adresse IP routable (ou un nombre réduit d'adresses IP routables) entre plusieurs machines dotées d'une adresse privée. Ainsi, vu de l'extérieur, toutes les machines du réseau interne possèdent la même adresse IP.

Les adresses IP privées sont donc la garantie d'une sécurité accrue. De plus, elles constituent une réponse au manque d'adresses IP. Leurs inconvénients sont : le travail supplémentaire lors de la configuration du réseau et la renumérotation à envisager lors de la fusion d'entreprises qui utiliseraient les mêmes adresses IP privées.

Les adresses sans classe CIDR (*Classless InterDomain Routing*)

Le CIDR a été proposé à partir de 1994. L'idée est d'organiser une adresse réseau indépendamment de sa classe ; le masque de sous-réseau indiquant le nombre de bits réservés à l'identifiant réseau est alors fixé librement par l'administrateur. Par exemple, pour réaliser l'équivalent de deux adresses de classe *C* contiguës, l'administrateur choisira un masque `/23`. Naturellement, le CIDR s'utilise également pour les adresses privées, telles qu'elles sont définies dans la RFC 1918.

Exemple

L'entreprise s'est vu attribuer l'adresse : `12.22.36.0 /22`. Cela veut dire que l'identifiant réseau tient sur 22 bits. Il reste 10 bits que l'administrateur peut affecter librement. Dans certaines parties du réseau, il peut décider d'utiliser un masque `/23`, dans d'autres il prendra un masque `/25` mais jamais un masque de taille inférieure à 22 bits dans notre exemple.

L'avantage de CIDR est de s'affranchir des contraintes imposées par le format des classes d'adresses. Les seules restrictions qui demeurent concernent les adresses dévolues au réseau lui-même, à la diffusion dans le réseau et aux anciennes classes *D* (réservée au multicast) et *E* (classe d'extension) d'IPv4.

Le tableau 6.2 donne des exemples d'allocation d'adresses en fonction des besoins de l'organisation (la liste est bien évidemment non exhaustive. Elle est à adapter aux besoins du réseau considéré). Nous donnerons d'autres exemples dans les compléments pédagogiques, sur le site www.pearsoneducation.fr.

Tableau 6.2

Exemples d'allocation d'adresses CIDR

Besoin de l'organisation	Allocation d'adresses	Masque utilisé
< 64 adresses	Subdivision de classe *C*	/26
< 128 adresses	Subdivision de classe *C*	/25
< 256 adresses	**1 réseau de classe C**	**/24**
< 512 adresses	2 réseaux de classe *C* contigus	/23
< 1 024 adresses	4 réseaux de classe *C* contigus	/22
< 2 048 adresses	8 réseaux de classe *C* contigus	/21
< 4 096 adresses	16 réseaux de classe *C* contigus	/20

Exemple

Soit une entreprise possédant 780 équipements dans son réseau qui obtient 4 adresses de classe *C* consécutives : `194.42.36.0`, `194.42.37.0`, `194.42.38.0` et `194.42.39.0`. Pour elle, tout se passe comme si son identifiant de réseau était `11000010 00101010 001001`, puisqu'il y a 22 bits en commun sur les identifiants réseau. En effet, dans le troisième octet, les nombres 36, 37, 38 et 39 ont en commun les 6 premiers bits `001001`. La notation de l'adresse IP d'un équipement quelconque dans ce réseau sera, par exemple, `194.42.37.156/22`, où `/22` (prononcer *slash* 22) signifie : « Dont les 22 premiers bits sont l'identifiant de réseau. » Sur les 32 bits de l'adresse, il en reste 32 – 22 = 10 pour l'identifiant de machine, ce qui permet 2^{10} adresses différentes et convient à cette entreprise.

La distribution dynamique des adresses

Une autre solution pour gérer la pénurie des adresses consiste à utiliser une plage d'adresses (éventuellement trop petite pour le parc de machines), en allouant temporairement les adresses IP disponibles aux seules machines connectées et en partant de l'hypothèse que toutes ne le seront pas simultanément. Pour assurer la distribution dynamique des adresses, le protocole DHCP (*Dynamic Host Configuration Protocol*) fournit automatiquement à un ordinateur qui vient d'être installé dans le réseau de l'entreprise ses paramètres de configuration réseau (adresse IP et masque de sous-réseau). De plus, cette technique sim-



plifie la tâche de l'administrateur d'un grand réseau, en évitant les doublons d'adresses. Elle peut se mettre en œuvre aussi bien avec des adresses publiques qu'avec des adresses privées et peut évidemment servir lorsque la plage d'adresses IP est plus grande que le parc de machines à identifier. Cette technique étant vue comme une application au sens de l'architecture de communication, nous l'aborderons au chapitre 9 qui traite des applications.

1.5 Association des adresses IP et des adresses MAC (Medium Access Control)

Au chapitre précédent, nous avons vu que les équipements étaient identifiés dans les réseaux locaux par leur adresse MAC, numéro de série de la carte réseau qui y est installée. Maintenant, les réseaux locaux sont interconnectés et « plongés » dans des réseaux logiques, qui attribuent à leurs équipements des adresses IP. Le même équipement possède donc deux adresses, l'une physique, l'autre logique, et il est nécessaire de pouvoir faire l'association entre les deux.

Soit deux machines *A* et *B*, connectées à un même réseau local. Chaque machine a une adresse IP, respectivement IP_A et IP_B, et une adresse MAC (numéro de série de la carte Ethernet, par exemple), respectivement MAC_A et MAC_B. Le problème, nommé « problème de résolution d'adresse ou *address resolution problem*, consiste à faire la correspondance entre adresses IP et adresses MAC, sachant que les programmes d'application ne manipulent – au mieux – que des adresses IP et que le réseau local ne voit que les adresses MAC. Dans chaque machine, des tables contiennent des paires adresse IP/adresse MAC, mais elles ne peuvent maintenir qu'un petit nombre de paires d'adresses. Un protocole de résolution d'adresses (ARP, *Address Resolution Protocol*) fournit un mécanisme efficace et simple. Il est défini dans la RFC 826.

Le protocole ARP (*Address Resolution Protocol*) [RFC 826]

Le protocole ARP établit une correspondance dynamique entre adresses physiques et adresses logiques. Il permet à une machine de trouver l'adresse physique d'une machine cible située sur le même réseau local, à partir de sa seule adresse IP. Lorsqu'une machine *A* veut résoudre l'adresse IP_B, elle diffuse un message spécial (en utilisant l'adresse MAC `FF:FF:FF:FF:FF:FF` comme identité de destinataire). Celui-ci demande à la machine d'adresse IP_B de répondre en indiquant son adresse physique MAC_B. Toutes les machines, y compris *B*, reçoivent ce message puisqu'il est envoyé en diffusion ; seule la machine *B* reconnaît son adresse IP. Elle répond en envoyant son adresse MAC_B. Lorsque *A* reçoit cette réponse, elle peut alors communiquer directement avec *B*. Les messages spéciaux que nous venons de voir, ceux du protocole ARP, sont véhiculés dans les données de la trame du réseau local. Un protocole similaire, baptisé RARP (*Reverse Address Resolution Protocol*) permet de la même façon, pour une machine sans disque, de connaître son adresse IP auprès d'un serveur d'adresses.

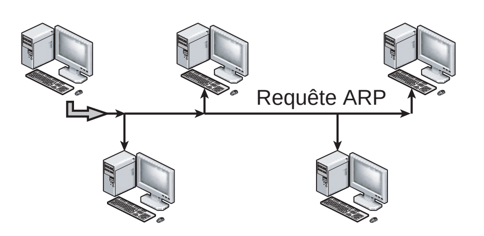
Le format de la requête/réponse ARP est très malléable car la taille des adresses n'est pas définie à l'avance et se trouve donc codée dans le message lui-même. La requête/réponse ARP contient (voir figure 6.2) :

- *L'adresse physique de l'émetteur.* Dans le cas d'une requête ARP, l'émetteur place son adresse ; dans une réponse ARP, ce champ révèle l'adresse recherchée.
- *L'adresse logique de l'émetteur* (l'adresse IP de l'émetteur).
- *L'adresse physique du récepteur.* Dans le cas d'une requête ARP, ce champ est vide.
- *L'adresse logique du récepteur* (l'adresse IP du récepteur).

Un mécanisme de cache permet de conserver les informations ainsi acquises : chaque système dispose d'une table pour sauvegarder les correspondances (adresse MAC, adresse IP). Ainsi, une requête ARP n'est émise que si le destinataire est absent dans la table.

Figure 6.2

Requête ARP en diffusion.



Exemple

La commande `arp  a` affiche le contenu de la table, aussi bien sous Windows que sous Unix :

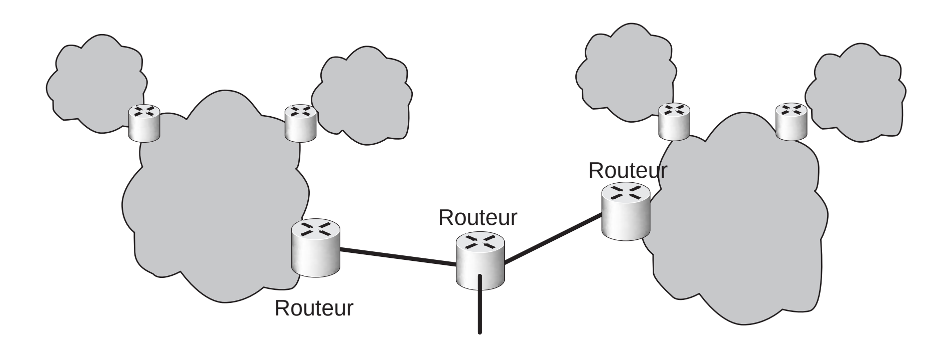
```
arp -a
poste_1(192.168.1.2) at 02:54:05:F4:DE:E5 [ether] on eth0
poste_2(192.168.1.1) at 02:54:05:F4:62:30 [ether] on eth0
```

2 Service rendu par le protocole IP

Un module logiciel gérant le protocole IP est installé dans tous les équipements du réseau local. Un équipement d'interconnexion, le *routeur*, gère l'accès au monde extérieur (voir figure 6.3). Pour bien comprendre le service rendu par le protocole IP, nous décrivons le principe de traitement d'un message au format IP (un *datagramme*) à travers l'interconnexion de réseau, en dehors de tout événement anormal.

Figure 6.3

Interconnexion de réseaux IP par des routeurs.



2.1 Création d'un datagramme par un équipement émetteur

Lorsqu'une application qui s'exécute sur l'équipement *A* veut transmettre des données à l'équipement *B*, elle donne un ordre au module IP de la machine *A*, traduit par une *requête d'émission* comprenant deux paramètres : les données et l'adresse du destinataire IP_B. Dans son fichier de configuration, le module IP de la machine *A* possède des informations comme : l'adresse IP_A, le masque de sous-réseau, l'adresse IP_R du routeur permettant de sortir du réseau. IP_R est considérée par l'équipement comme la route par défaut pour joindre tout destinataire extérieur au réseau. Le module IP de la machine *A* fabrique alors un datagramme, conformément au format défini par le protocole, puis il sollicite sa carte réseau pour que le datagramme soit encapsulé dans une trame du réseau local et transmis jusqu'au destinataire final (si *B* est dans le même réseau local) ou



jusqu'au routeur (si *B* est à l'extérieur du réseau). Le traitement de la requête d'émission est terminé pour la machine *A* : il n'y a aucun établissement de connexion entre *A* et *B*, aucun dialogue préalable entre eux. Le module IP gère les requêtes une par une, indépendamment les unes des autres. Aucun accusé de réception n'est prévu dans le protocole.

2.2 Traitement du datagramme par un routeur

Le routeur est, par définition, un équipement connecté à au moins deux réseaux IP. Il possède donc au moins deux interfaces réseaux (éventuellement différentes l'une de l'autre) et, de ce fait, a*u moins deux* adresses IP (en fait, il possède autant d'adresses IP que d'interfaces physiques différentes).

Lorsqu'un routeur reçoit sur l'une de ses interfaces réseau la trame qui lui est adressée, il la décapsule et en extrait le datagramme IP qu'elle contenait ; il lit alors l'adresse IP de destination (ici IP_B). Le rôle du routeur est de chercher dans sa *table de routage* la route qui permet d'atteindre le réseau de *B*. La table de routage contient la liste de réseaux connus ainsi que l'adresse du *routeur suivant* pour les atteindre (et l'indication locale de l'interface réseau concernée). À partir des informations de la table, il peut en déduire le chemin à suivre. Par exemple :

pour aller à	réseau *IP X*	passer par	routeur *R2*	en utilisant la carte d'interface	Ethernet2
pour aller à	réseau *IP Y*	passer par	routeur *R3*	en utilisant la carte d'interface	Ethernet2
pour aller à	réseau *IP Z*	passer par	routeur *R3*	en utilisant la carte d'interface	Ethernet1
pour aller à	ailleurs	passer par	routeur *R1*	en utilisant la carte d'interface	ls1

La table de routage contient toujours une entrée par défaut, dans le cas où le réseau du destinataire ne serait pas connu. Le routeur sollicite donc la carte d'interface spécifiée dans sa table et lui demande d'expédier le datagramme au routeur indiqué. Le datagramme est encapsulé dans une nouvelle trame, à destination d'un deuxième routeur, et ainsi de suite jusqu'au réseau du destinataire. Le routeur n'établit aucune connexion ni avec *A* ni avec le routeur suivant, il n'y a aucun dialogue préalable entre eux. Il gère les datagrammes, indépendamment les uns des autres. Aucun accusé de réception n'est prévu. Deux datagrammes qui se suivraient avec même émetteur et même destinataire peuvent ne pas suivre la même route et donc ne pas arriver au destinataire final (s'ils arrivent…), dans l'ordre où ils ont été émis : il suffit que la table de routage ait été mise à jour à la suite d'un incident par exemple.

2.3 Réception d'un datagramme par un équipement destinataire

Le datagramme traverse ainsi l'interconnexion de réseaux, de routeur en routeur, jusqu'à atteindre le routeur d'entrée du réseau de *B*. Encapsulé une dernière fois dans une trame du réseau local de *B*, il parvient à son destinataire. La carte réseau de la machine *B* reçoit cette dernière trame qui lui est adressée ; elle en sort le datagramme qu'elle livre à son module IP. Celui-ci analyse l'adresse et reconnaît la sienne : il signale alors à l'application concernée l'arrivée des données, ce qui se traduit par une *indication de réception* avec deux paramètres : les données et l'adresse de l'émetteur IP_A. Le traitement est terminé pour *B*.

2.4 CONCLUSION

Le protocole IP, implanté dans tous les équipements du réseau (machines et routeurs), assure un service de remise des données non fiable et sans connexion. Il comprend la définition du plan d'adressage, la structure des informations transférées (le datagramme IP) et les règles de routage. Nous avons vu que les datagrammes sont indépendants les uns des autres ; ils sont acheminés à travers l'interconnexion, en fonction des adresses IP publiques (source et destination). Les différents routeurs choisissent un chemin à travers les réseaux ; ils fragmentent les datagrammes lorsque le réseau suivant n'accepte que des messages de taille plus petite. La MTU (*Maximum Transfer Unit*) d'un réseau Ethernet est par exemple de 1 500 octets, celle d'un autre réseau peut être de 128 octets. Une fois le datagramme morcelé, les fragments sont acheminés comme autant de datagrammes indépendants jusqu'à leur destination finale où ils doivent être réassemblés.

Pour trouver un chemin jusqu'au destinataire, les routeurs s'échangent, dans des messages spéciaux, des informations de routage concernant l'état des différents réseaux. Ces informations sont véhiculées elles aussi par IP. Elles servent à mettre à jour les tables de routage qui indiquent, pour chaque identifiant de réseau, si les machines situées dans le réseau sont accessibles directement ou non. Le routage est *direct* si les machines appartiennent au même réseau, sinon il est *indirect*. Dans ce cas, le routeur émetteur envoie le datagramme au routeur suivant ; la coopération des deux routeurs permet de bien acheminer le datagramme. Différents protocoles comme GGP (*Gateway to Gateway Protocol*), EGP (*Exterior Gateway Protocol*), RIP (*Routing Information Protocol*), OSPF (*Open Shortest Path First*) sont utilisés entre les différents types de routeurs pour échanger et effectuer la mise à jour des informations de routage. Nous les aborderons au chapitre 8 consacré au routage.

L'envoi de messages d'erreur est prévu en cas de destruction de datagrammes, de problèmes de remise ou d'acheminement ; ces messages sont gérés par ICMP (*Internet Control Message Protocol*), un protocole que nous verrons à la section 4.

3 Format du datagramme IP

Le datagramme IP comprend un en-tête et des données. L'en-tête contient principalement les adresses IP de la source et du destinataire, et des informations sur la nature des données transportées (voir figure 6.4).

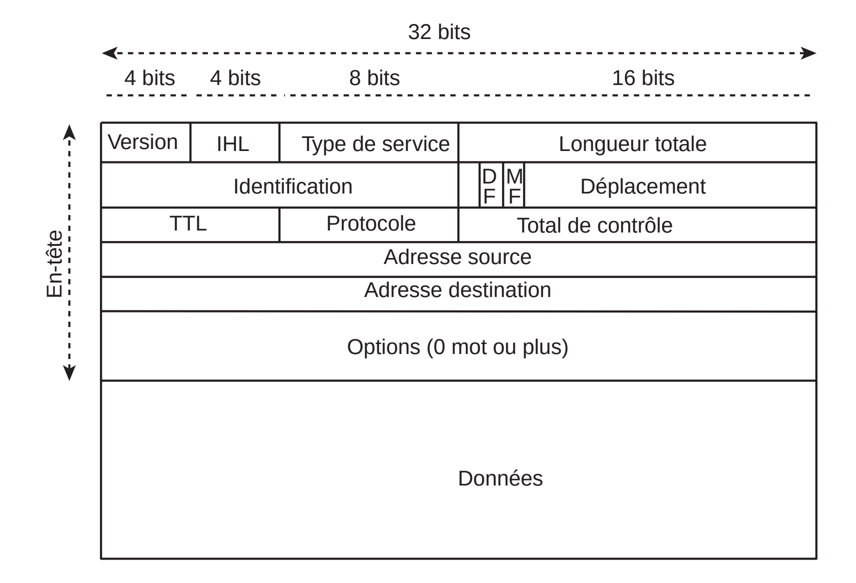
Classiquement, les différents champs sont décrits par des mots de 32 bits. La première ligne de la figure 6.4 contient quatre champs :

- *Version*. Il s'agit de la version du protocole IP qu'on utilise (actuellement, c'est la version 4 ou IPv4) afin de vérifier la validité du datagramme. La version est codée sur 4 bits.
- *Longueur de l'en-tête*. Le nombre de mots de 32 bits de l'en-tête (qui commence avec le champ version). La longueur est également codée sur 4 bits. De ce fait, un en-tête IP contient (en hexadécimal) au maximum F mots de 32 bits, soit 60 octets.
- *Type de services* (ToS). Ce champ de 8 bits indique la façon dont le datagramme doit être traité. Historiquement, il était possible de demander que le datagramme soit traité sur la route la plus rapide, sur celle qui offrait le meilleur débit, la plus fiable, etc. Encore fallait-il être capable de mesurer l'état des routes et de gérer les options… Les premières implémentations du protocole IP ont vite abandonné cette idée de services différenciés. Le champ ToS est resté à 0, d'autant que plusieurs propositions incompa-



Figure 6.4

Le datagramme IP.



tibles les unes avec les autres ont été faites pour modifier l'attribution de ce champ. Nous verrons à la section 5 que la version IPv6 reprend, sous une forme différente, l'idée de qualité de service.

- *Longueur totale*. Ce champ de 16 bits exprime en octets la taille totale du datagramme (en-tête + données). La longueur maximale d'un datagramme est donc 64 Ko, mais des raisons physiques imposent des tailles inférieures dans la plupart des réseaux.

Le deuxième mot de 32 bits concerne la fragmentation. Le champ *Identification* est un numéro de 16 bits attribué à chaque datagramme. Chaque fragment d'un même datagramme reprend le même identifiant, pour permettre le réassemblage correct du datagramme initial chez le destinataire. Après un premier bit non utilisé, les deux bits suivants sont des *drapeaux* qui permettent le réassemblage :

- *DF* : *Don't Fragment* (le deuxième bit). Autorise ou non la fragmentation du datagramme (si DF = 0 la fragmentation est autorisée, interdite si DF = 1). Par convention, toute machine doit pouvoir transmettre en un seul datagramme des données de 476 octets.
- *MF* : *More Fragments* (le dernier bit). Indique si le fragment de données est suivi ou non par d'autres fragments (si MF = 0, le fragment est le dernier du datagramme).

Le champ *Déplacement* permet de connaître la position du début du fragment par rapport au datagramme initial. Le fragment doit avoir une taille qui est un multiple entier de 8 octets. Le déplacement est codé sur les 13 derniers bits du mot.

Le troisième mot de 32 bits contient trois champs :

- *Durée de vie* (*TTL*, *Time To Live*). Indique sur 8 bits le nombre maximal de routeurs que le datagramme peut traverser. Ce champ était prévu à l'origine pour décompter un temps, d'où son nom. La durée de vie est choisie par l'émetteur ; elle est décrémentée chaque fois que le datagramme traverse un routeur. Lorsque la durée de vie atteint la valeur nulle, le datagramme est détruit.
- *Protocole.* Champ de 8 bits indiquant à quel protocole sont destinées les données véhiculées dans le datagramme. Les valeurs décimales les plus courantes sont : 1 pour

ICMP, 2 pour IGMP (*Internet Group Management Protocol*, ou protocole de gestion des groupes multicast), 6 pour TCP et 17 pour UDP.

- *Header checksum.* Ces 16 bits suivants constituent un bloc de contrôle d'erreur pour l'en-tête : ce champ permet de contrôler l'intégrité de l'en-tête. Celui-ci, en effet, transporte toutes les informations fondamentales du datagramme. Si, par hasard, il était détecté en erreur, le datagramme serait directement écarté. Remarquons qu'il n'y a aucune protection concernant les données transportées dans le datagramme.

Les deux derniers mots de 32 bits contiennent, dans cet ordre, l'*adresse IP source* et l'*adresse IP destination*. Ces cinq mots constituent l'en-tête minimal, commun à tous les datagrammes IP. En plus de ces informations, l'en-tête peut contenir en option des informations supplémentaires. C'est pourquoi il faut en indiquer la longueur.

Les options doivent tenir sur un nombre entier de mots de 32 bits. Parmi elles, le *routage* et l'*horodatage* sont particulièrement intéressantes (l'horodatage demande à chaque routeur d'estampiller le datagramme avec la date et l'heure à laquelle il a été traité). Elles constituent un bon moyen de surveiller ou de contrôler la traversée des datagrammes dans le réseau. Une autre option, l'*enregistrement de route*, demande à chaque routeur traversé de placer sa propre adresse dans le datagramme. Le destinataire reçoit ainsi un datagramme contenant la liste des adresses des routeurs traversés. Le *routage défini par la source*, lui, permet à l'émetteur d'imposer le chemin par lequel doit passer un datagramme.

3.1 Contrôle du datagramme dans un routeur

L'en-tête du datagramme est protégé par un bloc de contrôle d'erreur. Avant tout traitement, celui-ci est vérifié ; si des erreurs y sont détectées, le datagramme est détruit. D'autre part, le passage par un routeur provoque la diminution de 1 du champ durée de vie. Celui-ci faisant partie de l'en-tête, il est inclus dans le calcul du bloc de contrôle d'erreur. Le routeur l'ayant modifié, il doit donc recalculer le bloc de contrôle d'erreur avant de transférer le datagramme.

3.2 Gestion de la fragmentation

Maintenant que nous connaissons les détails du format d'un datagramme, examinons comment le fragmenter. Soit un réseau 1 où la MTU est 1 500 octets ; le routeur *R1* de ce réseau le relie à un réseau 2 de MTU égale à 576 octets. Un routeur *R2* le relie à un réseau 3 de MTU 1 500 octets (voir figure 6.5). La machine *A* du réseau 1 envoie un datagramme contenant 1 480 octets de données à la machine *B* située sur le réseau 3.

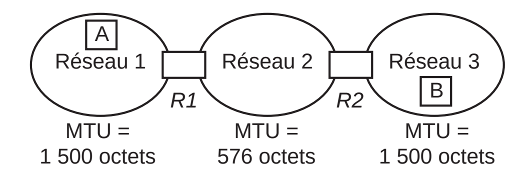


Figure 6.5

Nécessité de fragmentation dans un réseau intermédiaire.

Le datagramme fabriqué par *A* porte l'identification 17C9 (hexadécimal), le bit DF à 0 (fragmentation autorisée), le bit MF et le champ Déplacement à 0. Sachant que l'en-tête d'un datagramme contient 20 octets, le routeur *R1* va fragmenter le datagramme en trois morceaux pour le passer dans le réseau 2. Les deux premiers fragments contiendraient



556 octets de données s'il n'y avait la contrainte d'une taille multiple de 8 octets. Il faut donc choisir la taille la plus proche soit 552 octets. Le champ Déplacement vaut alors 552/8 = 69 pour le deuxième fragment puisque le déplacement est exprimé en bloc de 8 octets. Le dernier fragment n'en compte que 376. Les trois fragments auront chacun un en-tête de 20 octets qui est celui du datagramme initial, sauf pour le bit MF et le champ Déplacement, comme le montre le tableau 6.3.

Tableau 6.3

En-tête des différents fragments

	Identification	MF	Déplacement
Fragment 1	17C9	1	0
Fragment 2	17C9	1	69
Fragment 3	17C9	0	138

Tous les datagrammes issus d'une fragmentation deviennent des datagrammes IP comme les autres. Ils peuvent arriver à destination, éventuellement dans le désordre. IP doit faire le tri en utilisant les informations de l'en-tête pour faire le réassemblage. Au bout d'un certain temps, si un fragment manque toujours, la totalité du datagramme est considérée comme perdue. Puisque aucun mécanisme de contrôle n'est implémenté dans IP, la fragmentation est une source d'erreurs supplémentaire.

4 Protocole ICMP (*Internet Control Message Protocol*)

Dans l'interconnexion de réseaux, chaque routeur fonctionne de manière autonome. Des anomalies, dues à des pannes d'équipement ou à une surcharge temporaire, peuvent intervenir. Pour réagir correctement à ces défaillances (qu'il faut déjà connaître), le protocole de diagnostic ICMP (RFC 590) s'occupe de la transmission des messages de contrôle. Chaque équipement surveille son environnement et signale les événements anormaux.

Nous avons vu que pour contrôler le trafic dans le réseau, un champ, dans l'en-tête du datagramme, indique la *durée maximale de séjour* dans l'interconnexion ; chaque routeur traitant le datagramme décrémente la durée de vie. Lorsqu'elle vaut zéro, le datagramme est détruit et le routeur envoie un message d'erreur à l'émetteur du datagramme.

ICMP est donc un mécanisme de contrôle des erreurs au niveau IP[1]. Initialement prévu pour permettre aux routeurs d'informer les utilisateurs des erreurs de transmission, il n'est pas restreint à cette fonction puisque des échanges entre utilisateurs sont tout à fait possibles.

Chaque message ICMP traverse le réseau en tant que données d'un datagramme IP. La conséquence directe est que les messages ICMP sont traités comme les autres datagrammes à travers le réseau. On comprend donc mieux la nécessité d'indiquer, dans son entête, le contenu du datagramme IP par le champ *Protocole*, bien qu'ICMP ne soit pas considéré comme un protocole de niveau plus élevé que IP. En fait, si les messages doivent traverser plusieurs réseaux avant d'arriver à leur destination finale, IP est le seul protocole commun à l'interconnexion.

1. Habituellement, les protocoles contiennent eux-mêmes les mécanismes nécessaires pour signaler les erreurs qui sont de leur ressort. IP fait exception, et le protocole ICMP a été défini pour les situations d'anomalies.

Chaque message ICMP possède un type particulier pour caractériser le problème qu'il signale. L'en-tête ICMP sur 32 bits contient le *type* (code de l'erreur sur 8 bits), un champ d'information complémentaire selon le type (sur 8 bits) et un bloc de contrôle d'erreur sur 16 bits utilisant le même mécanisme de vérification que pour les datagrammes IP. De plus, les messages ICMP transportent des données qui sont en fait le début du datagramme à l'origine du problème.

L'utilitaire *ping* crée un message ICMP de type 8 (*Echo Request*) que la machine envoie à l'adresse IP spécifiée pour tester si ce dernier est opérationnel. Si tel est le cas, le destinataire répond par un message ICMP de type 0 (*Echo Reply*), en renvoyant les données contenues dans le message émis.

L'utilitaire *traceroute* exploite, quant à lui, les messages ICMP de type 11 (*Time Exceeded*), en envoyant des datagrammes IP dont le champ TTL est délibérément trop petit ; ces datagrammes sont écartés par l'un des routeurs de l'interconnexion. L'événement donne lieu à message d'erreur ICMP retourné à l'expéditeur.

Quand un routeur ne peut pas délivrer un datagramme, il envoie un message ICMP de type 3 (*Destination unreachable*) à l'émetteur. Dans ce cas, le champ d'information complémentaire précise si c'est le réseau, la machine, le protocole ou le port qui est inaccessible. Le même message ICMP de type 3 est utilisé lorsqu'un routeur doit fragmenter un datagramme et que celui-ci porte le bit DF à 1 (le champ d'information complémentaire spécifie le cas).

5 Protocole IPv6 (IP version 6)

La croissance exponentielle du nombre d'ordinateurs connectés à Internet pose de nombreux problèmes. Le plan d'adressage IP atteint un seuil de saturation, les adresses disponibles commencent à manquer. Par ailleurs, le protocole IP dans sa version 4, présente plusieurs défauts : nécessité de recalculer le bloc de contrôle de l'en-tête dans chaque routeur, de configurer les machines avec une adresse IP, un masque de sous-réseau et une route par défaut, sans parler de l'absence de sécurité : il n'y a aucun service pour assurer la confidentialité des données transmises, pour authentifier les adresses utilisées… Pour terminer, IPv4 est incapable de traiter de façon satisfaisante des flux audio ou vidéo ou des flux à contraintes temporelles fortes comme les jeux interactifs. Toutes ces raisons ont motivé le développement d'une nouvelle version d'IP, appelée IPv6, qui prévoit un nouveau plan d'adressage, un format différent pour le datagramme, la notion de qualité de service et des mécanismes de sécurité.

5.1 Plan d'adressage

IPv6 prévoit des adresses sur 128 bits, ce qui est gigantesque : on pourrait utiliser plusieurs millions d'adresses par m^2 sur terre, y compris dans les océans ! Les types d'adresses sont globalement conservés, sauf les adresses de diffusion (*broadcast*) qui sont remplacées par une généralisation du *multicast* (adressage multipoint).

On ne parle plus de classes d'adresses, et de nombreux nouveaux types, déterminés par un préfixe, existent. Le préfixe `0000 0000` binaire est utilisé pour la compatibilité avec les adresses IP classiques. L'adressage IPv6 résout non seulement le problème de la saturation des adresses mais il offre, en plus, de nouvelles possibilités comme la hiérarchisation à plusieurs niveaux ou l'encapsulation d'adresses déjà existantes, ce qui facilite leur résolution.



5.2 Nouveau format et qualité de service

IPv6 utilise un format de datagramme incompatible avec IP classique. Il se caractérise par un en-tête de base de taille fixe et plusieurs en-têtes d'extension optionnels suivis des données. Ce format garantit une souplesse d'utilisation et une simplicité de l'en-tête de base. Seize niveaux de priorité sont définis et respectés par les routeurs : le traitement des applications interactives et des transferts de fichiers vidéo peuvent alors être différents. Un identificateur de *flot* relie les datagrammes d'une même connexion applicative afin de leur garantir la même qualité de service[2]. L'utilisation combinée de la priorité et de l'identificateur de flot permet d'ajuster la qualité de service offerte par le routage aux besoins de l'application. Elle répond donc à la demande des nouvelles applications (temps réel, multimédia…).

Le nombre de routeurs que peut traverser le datagramme avant d'être détruit remplace le champ Durée de vie d'IPv4. Sa gestion est plus simple. La fragmentation est désormais traitée de bout en bout : l'algorithme PMTU (*Path Maximum Transfer Unit*) détermine la taille maximale des datagrammes sur le chemin prévu. Les paquets sont ensuite fragmentés par la source et réassemblés par le destinataire.

Grâce à l'utilisation d'en-têtes optionnels, le routeur n'a qu'à extraire l'en-tête de base ainsi que l'en-tête optionnel *hop by hop* (littéralement saut par saut) qui suit l'en-tête de base et qui contient des options devant être traitées dans les nœuds intermédiaires.

Avec le développement des portables, il est intéressant de pouvoir rediriger les messages adressés à la station fixe habituelle vers sa localisation actuelle en cas de déplacement. Cela se fait désormais au niveau du protocole IPv6 (et non au niveau des protocoles de couches supérieures, comme c'est le cas avec la redirection des courriers électroniques, par exemple). Un redirecteur, placé à l'entrée du réseau, connaît l'adresse IPv6 de la personne en déplacement. Il encapsule le datagramme dans un nouveau datagramme IPv6 et l'expédie à la nouvelle adresse. Le destinataire peut ainsi connaître l'identité de l'émetteur.

Lorsqu'il existe des contraintes de délai et de débit (temps réel), les routeurs mettent également en œuvre un mécanisme de réservation de ressources adapté aux exigences stipulées dans les champs priorité et identificateur de flot des datagrammes. Enfin, IPv6 implémente des éléments d'authentification et de confidentialité, thèmes qui n'étaient pas abordés dans IP, mais seulement dans la version IPSec utilisée pour créer des réseaux privés virtuels.

Résumé

Le protocole IP (*Internet Protocol*) est implémenté dans toutes les machines hôtes d'Internet ainsi que dans tous les routeurs. Il assure un service de remise non fiable sans connexion. Il comprend la définition du plan d'adressage, la structure de l'unité de données transférée (le *datagramme IP*) et des règles de routage. Les datagrammes sont transmis à travers l'interconnexion, au coup par coup, indépendamment les uns des autres. IP inclut un protocole ICMP de génération de messages d'erreur en cas de destruction de datagrammes, de problèmes d'acheminement ou de remise. La saturation du plan d'adressage, l'absence de qualité de service et de sécurité ont conduit à de nombreuses études dont est issu IPv6, la nouvelle version du protocole IP.

2. Mais IPv6 reste un protocole sans connexion !

Problèmes et exercices

Exercice 1 Principes généraux de l'adressage

Énoncé

a Quel est l'avantage de la séparation de l'adressage en deux parties dans l'adressage Internet ?

b Pourquoi l'adresse IP ne peut-elle pas être affectée à un périphérique réseau par son fabricant ?

Solution

a Le fait de séparer l'adresse en deux parties permet de réduire la taille mémoire des routeurs, qui ne conservent que l'adresse des (sous-)réseaux et celles des stations des (sous-) réseaux directement rattachées. En effet, la séparation entre l'adresse du réseau et celle de la station attachée au réseau permet un routage effectif dans les routeurs uniquement d'après l'adresse du réseau. L'adresse complète n'est utilisée qu'une fois le paquet arrivé dans le routeur connecté au réseau destinataire.

b L'adresse IP doit non seulement être unique mais elle doit aussi refléter la structure de l'interconnexion. La partie réseau de l'adresse dépend donc du réseau auquel est connectée la station : toutes les machines connectées au même réseau physique ont le même préfixe réseau.

Exercice 2 Classes d'adresse

Énoncé

L'adresse de ma machine est `193.48.200.49`. Puis-je en déduire si le réseau est de classe *A*, *B* ou *C* ?

Solution

`193` s'écrit en binaire `11000001` et commence donc par `110` : c'est une adresse de classe *C*.

Exercice 3 Informations de configuration

Énoncé

A et *B* sont deux utilisateurs de la même entreprise. L'utilisateur *A* a pour adresse `143.27.102.101` et lit dans le fichier de configuration de son poste (commande *ipconfig* ou *ifconfig*, par exemple) : masque de sous-réseau : `255.255.192.0` et adresse routeur par défaut : `143.27.105.1`.

a Quelle est l'adresse du sous-réseau auquel appartient *A* ? Quelle est l'adresse de diffusion sur ce sous-réseau ?

b L'utilisateur *B* a pour adresse `143.27.172.101` et lit de même : masque de sous-réseau : `255.255.192.0`. *B* est-il sur le même sous-réseau que *A* ? Peut-il utiliser la même adresse de routeur par défaut que *A* ?

Solution

a *A* est dans le réseau `143.27.0.0`, dans le sous-réseau `143.27.64.0` (on obtient `64` en faisant le ET entre les nombres `102` et `192` écrits sur 8 bits soit `01100110` ET `11000000`. Le résultat donne : `01000000` = `64`). Il y a donc 2 bits pour définir les sous-réseaux. L'adresse de diffusion dans ce sous-réseau est `143.27.127.255` (on obtient `127.255` en remplaçant les 14 bits prévus pour l'identifiant de machine par des 1).



b L'utilisateur *B* est dans le réseau `143.27.0.0` mais pas dans le même sous-réseau (il est dans le sous-réseau `143.27.128.0`). Il ne peut donc pas utiliser la même adresse de routeur par défaut (le routeur par défaut est obligatoirement dans le sous-réseau de l'utilisateur).

Remarque

Dans ce réseau, il n'y a qu'un routeur possédant deux interfaces internes et une interface vers le monde extérieur. Les deux utilisateurs *A* et *B* utilisent le même routeur pour transmettre des messages entre eux ou vers l'extérieur. Chaque utilisateur désigne le routeur par l'adresse IP de l'interface réseau qu'il connaît. On voit donc bien que l'adresse IP ne définit pas une machine mais une interface réseau.

Exercice 4 Adresse MAC et adresse IP

Énoncé

Soit une entreprise disposant d'un réseau Ethernet relié à Internet. Elle dispose d'une adresse IP de classe *B*. Son identifiant réseau est égal à `29C2` (en hexadécimal). Sur le réseau, il y a déjà deux cents ordinateurs dont l'adresse IP a été choisie dans l'ordre croissant en commençant par 1. Vous branchez un nouvel ordinateur disposant d'une carte Ethernet d'adresse MAC `3E:98:4A:51:49:76`.

a Proposez une adresse IP pour l'ordinateur et représentez-la sous forme décimale pointée.

b L'ordinateur est déplacé vers le réseau Ethernet d'une autre entreprise, ce réseau étant également connecté à Internet. Est-il nécessaire de changer l'adresse de la carte Ethernet ? De changer l'adresse IP de l'ordinateur ?

Solution

a L'adresse IP est de classe *B* donc commence par `10`. L'identifiant réseau s'écrit sur 14 bits : `29C2` soit `10 1001 1100 0010` en binaire. Donc la partie réseau vaut : `1010 1001 1100 0010` c'est-à-dire `169.194` en décimal. L'identité de la machine pourrait valoir `201` (en décimal). Son adresse IP serait alors `169.194.0.201`.

b L'adresse de la carte Ethernet est gérée dans la sous-couche MAC, comme son nom l'indique. Il n'est pas nécessaire d'en vérifier l'unicité. Celle-ci est garantie par le constructeur. Au niveau international, chaque constructeur a son préfixe et numérote ensuite chacune de ses cartes dans l'absolu. Par définition de l'adressage Ethernet, la carte conserve son adresse même quand l'ordinateur change de réseau. Par contre, il faut lui donner une nouvelle adresse IP correspondant au nouvel identifiant réseau et, éventuellement, une nouvelle identité de machine dans ce réseau (si une machine du nouveau réseau la possède déjà).

Exercice 5 Correspondance adresse MAC/adresse IP

Énoncé

Considérons deux machines, *1* et *2*, reliées au même réseau local. Chaque machine a une adresse IP, respectivement notées IP_1 et IP_2, et une adresse MAC, respectivement notées PH_1 et PH_2.

a Comment la machine *1* désirant envoyer un datagramme IP vers la machine *2*, dont elle ne connaît que l'adresse IP_2, peut-elle mettre en correspondance l'adresse IP_2 avec l'adresse physique PH_2 ?

b La machine *2* se trouve sur un réseau local distant totalement différent du précédent ; il est accessible à travers Internet. Comment le datagramme est-il transmis dans le réseau local de la machine *1* ? Quelles adresses portent la trame qui le transporte ? D'où viennent-elles ?

Solution

a La machine *1* doit rechercher l'adresse MAC de la machine *2* qu'elle connaît à travers IP_2. Elle consulte sa table ARP. Si celle-ci contient l'information, le problème est résolu. Si elle ne contient pas l'information, la machine *1* doit utiliser le protocole ARP en diffusant dans le réseau local une trame qui contient la requête ARP suivante : « Je cherche l'adresse MAC de la machine dont je connais l'adresse IP_2. » La trame étant diffusée, tous les équipements du réseau local la reçoivent. Seul l'équipement concerné par l'échange, c'est-à-dire ici la machine *2*, répond par une trame contenant la réponse ARP suivante : « Je suis la machine IP_2, mon adresse physique est PH_2. » En recevant cette réponse, la machine *1* met à jour sa table ARP en lui ajoutant une nouvelle ligne où IP_2 correspond à PH_2.

b Si la machine *2* est sur un autre réseau local, elle possède une adresse IP_2 qui n'appartient pas au même réseau que IP_1 (la machine *1* le sait en utilisant le masque de sous-réseau). Le datagramme doit être acheminé à l'extérieur du réseau ; il est envoyé localement à l'équipement qui assure ce service, c'est-à-dire au routeur. L'adresse IP_R du routeur est présente dans le fichier de configuration de la machine *1*. Dans le cas où la machine *1* ignore l'adresse physique PH_R du routeur, il lui faut rechercher cette adresse au moyen d'une requête ARP comme à la question a. Ensuite, la machine *1* émet dans le réseau local une trame dont les adresses physiques sont destinataire = PH_R et émetteur = PH_1. Cette trame encapsule un datagramme IP dont les adresses logiques sont émetteur = IP_1 et destinataire = IP_2.

Exercice 6 Sous-réseaux

Énoncé

Complétez le tableau :

Adresse IP	124.23.12.71	124.12.23.71	194.12.23.71
Masque de sous-réseau	255.0.0.0	255.255.255.0	255.255.255.240
Classe			
Adresse du réseau auquel appartient la machine			
Adresse de diffusion dans le réseau			
Adresse du sous-réseau auquel appartient la machine			
Adresse de diffusion dans le sous-réseau de la machine			

Solution

Adresse IP	124.23.12.71	124.12.23.71	194.12.23.71
Masque de sous-réseau	255.0.0.0	255.255.255.0	255.255.255.240
Classe	*A*	*A*	*C*
Adresse du réseau auquel appartient la machine	124.0.0.0	124.0.0.0	194.12.23.0
Adresse de diffusion dans le réseau	124.255.255.255	124.255.255.255	194.12.23.255
Adresse du sous-réseau auquel appartient la machine	pas de sous-réseau	124.12.23.0	194.12.23.64
Adresse de diffusion dans le sous-réseau de la machine		124.12.23.255	194.12.23.79



EXERCICE 7 PLAN D'ADRESSAGE GÉNÉRAL

Énoncé

Un site local est composé de deux sous-réseaux physiques reliés au reste du monde par le même routeur. Ce site possède une adresse IP de classe *B*. Proposez un mode d'adressage des différentes stations sur le site pour que le routeur n'ait pas à diffuser systématiquement tous les messages reçus du reste du monde dans les deux sous-réseaux.

Solution

Adresse de classe *B* : `x.y.0.0` avec *x* compris entre `128` et `191`. En absence d'hypothèse précise sur le nombre de machines dans chaque sous-réseau et sur l'évolution future du réseau, on considère qu'il suffit de créer deux sous-réseaux (ce qui nécessite 2 bits si on veut éviter les sous-réseaux « plein 0 » et « plein 1 »), donc un masque `255.255.192.0`. Dans les adresses IP des stations, les 16 premiers bits représentent le réseau (`x.y.`), les deux bits suivants les sous-réseaux (`01` et `10`). Les 14 bits restants désignent la machine elle-même.

Le sous-réseau `01` a pour adresse de sous-réseau `x.y.64.0` ; les adresses des machines vont de `x.y.64.1` à `x.y.127.254` ; l'adresse de diffusion dans ce sous-réseau est `x.y.127.255`. Tout message parvenant au routeur avec une adresse IP dans l'intervalle ci-dessus est diffusé exclusivement dans ce sous-réseau.

Le sous-réseau `10` a pour adresse de sous-réseau `x.y.128.0` ; les adresses des machines vont de `x.y.128.1` à `x.y.191.254` ; l'adresse de diffusion dans ce sous-réseau est `x.y.191.255`. Tout message parvenant au routeur avec une adresse IP dans l'intervalle ci-dessus est diffusé exclusivement dans ce sous-réseau.

EXERCICE 8 PLAN D'ADRESSAGE PARTICULIER

Énoncé

Une société veut se raccorder à Internet. Pour cela, elle demande une adresse réseau de classe *B* afin de contrôler ses 2 853 machines installées en France.

a Une adresse réseau de classe *B* sera-t-elle suffisante ?

b L'organisme chargé de l'affectation des adresses réseau lui alloue plusieurs adresses de classe *C* consécutives au lieu d'une adresse de classe *B*. Combien d'adresses de classe *C* faut-il allouer à cette société pour qu'elle puisse gérer tous ses terminaux installés ?

c Finalement, la société a pu obtenir une adresse réseau de classe *B*. L'administrateur du réseau choisit de découper le réseau pour refléter la structure de la société, c'est-à-dire qu'il crée autant de sous-réseaux que la société compte de services différents. L'administrateur a donc prévu 12 sous-réseaux, numérotés de 1 à 12. Proposez le masque de sous-réseau utilisé dans l'un des services de la société.

d Combien reste-t-il de bits pour identifier les machines de chaque service ? Combien de machines peut-on identifier dans chaque service ?

e L'adresse réseau de la société est : `139.47.0.0`. Indiquez l'adresse réseau du sous-réseau n° 9.

f Dans le sous-réseau choisi, donnez l'adresse IP complète de la machine ayant comme identifiant de machine `7.48`.

g Donnez les adresses réseau et les adresses de diffusion du sous-réseau n° 12.

Solution

a Oui, car une adresse de classe *B* permet d'adresser $2^{16} - 2$ (65 534 machines), soit largement plus que le nombre de machines installées.

b Une adresse de classe *C* permet d'adresser 254 machines. Il faut 12 adresses de classe *C* pour adresser tous les terminaux.

c Il faut 4 bits pour identifier 12 sous-réseaux. Le masque vaut donc : `255.255.240.0`.

d Il reste 12 bits, c'est-à-dire qu'on peut adresser $2^{12} - 2$ machines soit 4 094 machines par sous-réseau.

e Le sous-réseau n° 1 a pour adresse `139.47.16.0` (les 4 bits de sous-réseau valent `0001` soit 1 en décimal) donc le sous-réseau n° 9 aura pour adresse réseau : `139.47.144.0` (les 4 bits de sous-réseau valent `1001` soit 9 en décimal)

f La machine `7.48` du sous-réseau `139.47.144.0` a pour adresse IP `139.47.151.48`.

g Adresse réseau du sous-réseau n° 12 : `139.47.192.0` ; son adresse de diffusion vaut : `139.47.207.255`.

EXERCICE 9 PLAN D'ADRESSAGE AVEC SOUS-RÉSEAUX

Énoncé

Dans un réseau local Ethernet 100 Mbit/s, on dispose de 50 machines d'utilisateurs, réparties en 5 groupes de 10 machines et de 7 serveurs, à raison d'un serveur spécifique dans chaque groupe et de 2 serveurs communs à l'ensemble des utilisateurs. Dans chacun des 5 groupes, les machines des utilisateurs sont reliées à un concentrateur 12 ports.

a L'entreprise possède l'adresse IP `193.22.172.0`. Peut-on répartir les adresses en faisant apparaître les 5 groupes ? Si oui, comment ? Proposez un plan d'adressage.

b Soit un routeur d'entreprise qui relie 4 sous-réseaux *RL1*, *RL2*, *RL3* et *RL4* et offre l'accès à Internet. L'entreprise a une adresse IP de classe *C*, d'identité réseau égale à `195.52.100.0`. Dans le sous-réseau *RL1*, il y a 15 postes de travail, dans *RL2* 20 postes, *RL3* 25 postes, *RL4* 30 postes. Peut-on imaginer un plan d'adressage avec 4 sous-réseaux distincts ? Quel sera alors le masque de sous-réseau ?

Solution

a L'entreprise dispose de 50 + 7 = 57 machines : une adresse de classe *C* lui suffit. Pour faire apparaître 6 sous-réseaux (un par groupe et un pour les deux serveurs communs), il faut au moins 3 bits. Il reste alors 5 bits soit 32 – 2 = 30 adresses disponibles, ce qui convient parfaitement puisqu'il y a au maximum 11 postes par groupe. Le masque de sous-réseau sera `255.255.255.224`. Les 5 groupes d'utilisateurs correspondent aux sous-réseaux `193.22.172.32`, `193.22.172.64`, `193.22.172.96`, `193.22.172.128` et `193.22.172.160`. Les 2 serveurs seront dans le dernier sous-réseau `193.22.172.192`.

b Dans cet exemple, il faut faire 4 sous-réseaux ; on prendra 3 bits pour identifier les sous-réseaux. Les groupes sont de tailles différentes mais tous comptent au plus 30 postes. 5 bits pour identifier une machine sont suffisants. On pourra utiliser le même masque.

> **Remarque**
>
> Avec CIDR, on pourrait très bien se contenter de 2 bits pour identifier les 4 sous-réseaux, qui seraient alors numérotés de `0` à `3` (on aurait un masque /26). Dans ce cas, le réseau global et le sous-réseau 3 auraient la même adresse de diffusion : `193.22.172.255`.

Exercice 10 CIDR

Énoncé

Une entreprise cherche à obtenir une adresse publique alors qu'elle dispose d'un parc de 800 machines. Elle prévoit de posséder de 850 à 900 machines dans les années à venir. Elle obtient comme adresse réseau : `193.33.32.0/22`. Cette adresse lui convient-elle ? Quel est le masque de sous-réseau par défaut ?

Solution

L'indication /22 signifie que les 22 premiers bits sont dévolus à l'adresse réseau et que l'entreprise est libre d'utiliser les 10 bits restants pour identifier ses machines. Elle dispose donc d'un millier d'adresses, ce qui lui convient parfaitement.

Le masque de sous-réseau par défaut est alors, en découpant les octets : `11111111 11111111 11111100 00000000`, soit en décimal : `255.255.252.0`.

Exercice 11 Fragmentation des datagrammes

Énoncé

Un datagramme IP peut être segmenté en plusieurs fragments.

a De quelles informations dispose-t-on pour savoir qu'un datagramme contient un fragment ?

b Comment reconstitue-t-on un datagramme à l'arrivée ?

c Un routeur peut-il confondre deux fragments qui ont les mêmes éléments suivants : source, destination et numéro de fragment ?

Solution

a Le bit MF (*More Fragments*) est à 1 dans tous les fragments sauf le dernier ; le champ Déplacement n'est pas nul, sauf dans le premier fragment, alors qu'un datagramme non fragmenté possède un bit MF à 0 *et* un champ Déplacement à 0.

b Tous les fragments portent le même identificateur (celui du datagramme initial). On utilise alors le champ Déplacement pour reconstituer le datagramme. Le bit MF est à 0 dans le dernier fragment, à 1 dans tous les autres.

c Un routeur ne peut pas confondre deux fragments qui auraient les mêmes éléments source, destination et place de fragment, car le champ Identifiant du datagramme est forcément différent !

Exercice 12 Interconnexion

Énoncé

Deux sociétés *S1* et *S2* situées à 100 km l'une de l'autre fusionnent et désirent mettre en commun leurs moyens informatiques. La société *S1* possède un réseau Ethernet *E1* à 100 Mbit/s. Les transferts de données utilisent TCP/IP, avec une adresse IP de classe *C*. La société *S2* possède un réseau Ethernet *E2* à 100 Mbit/s sous TCP/IP, avec une adresse IP de classe *B*. Quelle est la structure de l'équipement d'interconnexion pour passer de *E1* à *E2* ? Quels sont les problèmes potentiels dus à l'interconnexion ?

Solution

Les deux sociétés étant éloignées, elles peuvent être reliées soit par une liaison spécialisée directe entre les deux sites soit par leur fournisseur d'accès à Internet. Dans les deux cas, il faut un routeur sur chaque site. Les problèmes principaux sont dus au fait que le réseau intermédiaire (la liaison spécialisée ou Internet) aura, selon toute vraisemblance, un débit inférieur à celui des deux réseaux Ethernet. Il faut donc qu'il n'y ait qu'un trafic limité entre les deux sites. La différence de classe des adresses n'a aucun impact sur les performances. Toutefois, le parc de l'ensemble des deux sociétés doit être suffisant pour que seule l'adresse de classe *B* soit utilisée. Il doit être possible de prévoir un plan d'adressage avec plusieurs sous-réseaux, dont un pour la société *S1*.

Remarque

L'idéal serait d'obtenir du fournisseur d'accès à Internet un service de réseau privé virtuel (voir compléments pédagogiques, sur le site www.pearsoneducation.fr) qui procure à l'entreprise l'illusion qu'elle dispose d'un réseau unique et sécurisé sur Internet.

Exercice 13 Répéteur, pont et routeur

Énoncé

Établissez un tableau comparatif entre les équipements d'interconnexion (répéteur, pont et routeur) en abordant les aspects suivants : niveau d'interconnexion, filtrage d'adresses, filtrage des collisions, filtrage du trafic de diffusion, génération de trafic de gestion, dépendance vis-à-vis des protocoles de communication, évolutivité, performances, impact sur la sécurité du réseau, reconfiguration, coût, temps de traitement interne, simplicité d'installation et de maintenance…

Solution

Éléments de comparaison	Répéteur	Pont	Routeur
Simplicité d'installation et de configuration	Oui	Oui	Non
Niveau d'inter-connexion	1	2	3
Filtrage	Non	Sur les adresses MAC	Sur les adresses IP
Trafic de service	Non	non sauf *Spanning Tree Algorithm*	Important *Routing Information Protocol*
Suppression des collisions	Non	Oui	
Protocoles traités	Aucun	Indépendant des protocoles	Une version de logiciel par protocole traité
Temps de traitement interne	Nul	Faible	Important
Évolutivité	Aucune	Faible	Grande
Filtrage du trafic de diffusion	Non	Non	Oui
Gestion de la sécurité du réseau	non	Non	Oui



EXERCICE 14 UTILITAIRE PING

Énoncé

Vous avez tapé dans une fenêtre de commande : `ping c1 193.93.28.7` alors que la machine cible est dans votre réseau. Vous obtenez en réponse (par exemple) :

```
64 bytes from 192.93.28.7. icmp_seq=0 ttl=255 time =0.7 ms
1 packet transmitted, 1 packet received, 0 % packet loss
round-trip (ms) min/avg/max = 0.7/0.7/0.7
```

La machine d'adresse IP `193.93.28.7` est-elle opérationnelle ? Que pensez-vous du délai de traversée du réseau ? Que signifie `icmp seq` ? Le message que vous avez envoyé a-t-il traversé un routeur ?

Solution

La machine d'adresse IP `193.93.28.7` est opérationnelle puisqu'elle a répondu à la requête d'écho. C'est là le rôle initial de l'utilitaire *ping* : tester si un équipement fonctionne en lui envoyant un message qu'il doit immédiatement renvoyer. Le délai de traversée est très bref, puisqu'on est à l'intérieur d'un réseau local. L'utilitaire *ping* envoie des messages du protocole ICMP (*Echo Request*) qui sont numérotés. Ici, il n'y en a qu'un, donc son numéro de séquence est `icmp seq` = 0. Le message n'a traversé aucun routeur puisque le champ TTL est à 255, ce qui représente sa valeur maximale.

EXERCICE 15 COMMANDE TRACEROUTE

Énoncé

Vous avez lancé une commande *traceroute* (*tracert* sous Windows). Cette commande permet de connaître le chemin suivi par un datagramme entre votre machine et une machine de destination spécifiée dans la commande. Vous avez obtenu le résultat suivant (voir tableau 6.4) :

Tableau 6.4

Commande traceroute

1	193.51.91.1	1 ms	1 ms	1 ms
2	2.0.0.1	23 ms	23 ms	23 ms
3	11.6.1.1	105 ms	35 ms	35 ms
4	11.6.13.1	37 ms	35 ms	34 ms
5	189.52.80.1	37 ms	60 ms	36 ms
6	193.48.58.41	51 ms	39 ms	46 ms
7	193.48.53.49	39 ms	47 ms	44 ms
8	193.220.180.9	44 ms	*	*
9	195.48.58.43	48 ms	38 ms	44 ms
10	195.48.58.50	145 ms	170 ms	64 ms
11	194.206.207.18	61 ms	146 ms	44 ms
12	194.207.206.5	166 ms	261 ms	189 ms

a Pourquoi le délai est-il au plus égal à 1 milliseconde pour la première ligne ?

b Que peuvent signifier les étoiles ?

c Comment expliquez-vous que pour la même destination les délais varient ?

d Combien de réseaux différents ont été traversés ?

e Peut-on connaître les protocoles utilisés ?

Solution

a La première ligne correspond au réseau local dans lequel se trouve l'utilisateur, le premier datagramme avec une durée de vie 1 a été détruit par le routeur de sortie du réseau. Il est donc normal que le délai soit très faible.

b Les étoiles correspondent à des datagrammes qui se sont perdus, à l'aller ou au retour : au-delà d'un certain délai, on les considère comme manquants.

c Les délais varient car rien n'est garanti dans l'interconnexion de réseaux : il peut y avoir des « embouteillages » momentanés et/ou des pannes qui provoquent des changements de route.

d Pour connaître le nombre de réseaux traversés, il suffit de calculer l'adresse réseau de chaque routeur et de compter le nombre de réseaux différents. Il y en a 10, comme le montre le tableau 6.5.

Tableau 6.5

Les réseaux traversés

193.51.91.1	193.51.91.0 (réseau 1)
2.0.0.1	2.0.0.0 (réseau 2)
11.6.1.1	11.0.0.0 (réseau 3)
11.6.13.1	11.0.0.0 (réseau 3)
189.52.80.1	189.52.0.0 (réseau 4)
193.48.58.41	193.48.58.0 (réseau 5)
193.48.53.49	193.48.53.0 (réseau 6)
193.220.180.9	193.220.180.0 (réseau 7)
195.48.58.43	195.48.58.0 (réseau 8)
195.48.58.50	195.48.58.0 (réseau 8)
194.206.207.18	194.206.207.0 (réseau 9)
194.207.206.5	194.207.206.5 (réseau 10)

e On ne peut pas connaître les protocoles utilisés au-delà de IP.

Exercice 16 Décodage de datagramme

Énoncé

Décoder le datagramme IPv4 suivant (en hexadécimal[3]) et en extraire toutes les informations possibles.

```
45 00 00 50 20 61 00 00 80 01 C5 64 C7 F5 B4 0A C7 F5 B4 09
08 00 00 1C 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10
11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24
25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38
```

3. La numération hexadécimale, qui a l'intérêt d'être compacte, est une représentation courante pour les données binaires : 4 bits sont représentés par un seul « chiffre » hexadécimal, dont les valeurs vont de 0 à F.



45 → 4 = protocole IP version 4 ; 5 = longueur de l'en-tête du datagramme = 5*4 = 20 octets = longueur par défaut d'un en-tête sans option.

00 → Type Of Service = 0 = pas de service particulier (en fait avec IPv4, il n'y a pas de service particulier. Ce champ est donc toujours nul !).

00 50 → longueur totale = 0*4096 + 0*256 + 5*16 + 0*1 = 80 octets donc la longueur du contenu du champ de données est de 80 – 20 = 60 octets.

20 61 → identificateur du datagramme (ne sera utile que s'il est fragmenté).

00 00 → drapeaux et déplacement = tout à zéro = datagramme non fragmenté.

80 → durée de vie = 80 = 8*16 + 0*1 = 128 routeurs que le datagramme pourrait encore traverser.

01 → protocole transporté dans le datagramme : 1 = code du protocole ICMP.

C5 64 → Bloc de contrôle d'erreur de l'en-tête.

C7 F5 B4 0A → adresse IP émetteur = 199.245.180.10.

C7 F5 B4 09 → adresse IP destinataire = 199.245.180.9.

Les deux machines sont dans le même réseau de classe C, le réseau 199.245.180.0.

-------Fin de l'en-tête IP---------------------

Pour décoder le contenu du datagramme, il faut connaître le format d'un message ICMP.

08 → type : 8

00 → code : 0

L'ensemble type = 8 et code = 0 signifie demande d'écho (couramment appelée *ping*).

00 1C → bloc de contrôle d'erreur sur l'en-tête du message ICMP.

-----Fin de l'en-tête ICMP--------------

Contenu quelconque destiné à être renvoyé par le destinataire s'il répond à cette demande d'écho :

```
01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10
11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20
21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30
31 32 33 34 35 36 37 38
```

Longueur du contenu ICMP = 56 octets.

---------Fin du contenu ICMP------

--------- Fin du contenu IP-------------

Bilan

Le datagramme est au format IPv4. Il a été émis par la machine d'adresse IP 199.245.180.10 vers la machine d'adresse IP 199.245.180.9. Ces deux machines sont dans le même réseau de classe *C*, le réseau 199.245.180.0. Le datagramme possède une longueur totale de 60 octets. Il transporte une requête ICMP de demande d'écho dont la longueur du contenu est de 56 octets : l'émetteur envoie un *ping* au récepteur pour connaître son état.

Exercice 17 Décodage de trame Ethernet

Énoncé

Décoder la trame Ethernet suivante (hexadécimal) et en extraire toutes les informations possibles.

```
AA AA AA AA AA AA AA AB 08 00 02 4B 01 C3 08 00 02 4B 02 D6 08 00 45 00
00 50 20 61 00 00 80 01 C5 64 C7 F5 B4 0A C7 F5 B4 09 08 00 00 1C 01 02
03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A
1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32
33 34 35 36 37 38 5F A6 8C 04
```

Solution

---------------------------- Début d'une trame Ethernet -------------------------------------

AA AA AA AA AA AA AA AB→ Synchronisation.

08 00 02 4B 01 C3→ *@MAC* destinataire (constructeur = 080002 = 3Com).

08 00 02 4B 02 D6→ *@MAC* émetteur (même constructeur).

08 00→ Type (ici *IP*). Si < à 1500 c'est une longueur.

[ici 08 00 = 2048, cette valeur ne peut donc pas être la longueur des données de la trame].

---------------------------- 46 <= contenu (ici datagramme IP) <= 1500 -------------------

Le contenu de cette trame est le *ping* de l'exercice précédent.

-------------------------------------Fin du contenu---

5F A6 8C 04 → Bloc de contrôle d'erreur Ethernet.

Bilan

Cette trame Ethernet a été capturée dans le réseau de classe *C* 199.245.180.0. Deux machines sont concernées par cet échange : la machine *X* d'adresse MAC 08 00 02 4B 02 D6 et d'adresse IP 199.245.180.10 qui a envoyé une requête d'écho (*ping*) à la machine *Y* d'adresse MAC 08 00 02 4B 01 C3 et d'adresse IP 199.245.180.9, située sur le même réseau local. Les cartes Ethernet sont du même constructeur. Les protocoles utilisés sont IP et ICMP.

Exercice 18 Autre décodage de trame Ethernet

Énoncé

Décodez la trame Ethernet suivante (en hexadécimal), dépourvue de son préambule de synchronisation et de son bloc de contrôle d'erreur. Extrayez-en toutes les informations possibles.

```
FF FF FF FF FF FF 00 04 80 5F 68 00 08 06 00 01
08 00 06 04 00 01 00 04 80 5F 68 00 89 C2 A2 03
00 00 00 00 00 00 89 C2 A2 F3 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00
```



Solution

---------------------------- Début d'une trame Ethernet -------------------------------------

FF FF FF FF FF FF → Adresse MAC destinataire (diffusion).

00 04 80 5F 68 00 → Adresse MAC émetteur.

08 06 → Type (ici ARP). Un nombre < 1500 donne la longueur des données de la trame (ici : 08 06 = 2054, ce ne peut donc pas être une longueur).

---------------------------- 46 ≤ Contenu (ici message ARP) ≤ 1500 -------------------------

Pour interpréter le contenu de cette trame, il faut disposer du format d'un message ARP.

00 01 → Type de matériel : 1 = Ethernet.

08 00 → Type de protocole : IP.

06 → Longueur de l'adresse physique : 6 octets (pour Ethernet).

04 → Longueur de l'adresse logique : 4 octets (pour IP).

00 01 → Code opération : 1 = Requête ARP.

00 04 80 5F 68 00 → Adresse MAC source.

89 C2 A2 03 → Adresse IP source : 137.194.162.3.

00 00 00 00 00 00 → Adresse MAC destination (vide car c'est l'adresse qu'on cherche).

89 C2 A2 F3 → Adresse IP destination : 137.194.162.243.

---------------------------- Fin du contenu réel--

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
bourrage (le contenu de la trame est trop court).

---------------------------- Fin du contenu (46 octets)--

Bilan

Dans un réseau de classe *B* 137.194.0.0, deux machines sont concernées par cette trame : les machines d'adresse IP 137.194.162.3 et 137.194.162.243. La machine 137.194.162.3 envoie une requête ARP en diffusion pour demander l'adresse physique de la machine d'adresse IP 137.194.162.243. La requête ARP est un message court (trop court pour une trame Ethernet) : elle est donc complétée avec 18 octets de bourrage.

Remarque

Le travail de décodage peut être fait par un équipement particulier appelé *analyseur de protocole*. Configuré avec les types de protocoles utilisés, l'analyseur fournit les informations brutes, découpées champ par champ. Le bilan, quant à lui, est le résultat de la réflexion humaine !

Chapitre 7

Les protocoles de transport

Le service fourni par IP n'étant pas fiable, il faut implanter par-dessus un protocole supplémentaire, en fonction de la qualité de service dont les applications ont besoin. Pour les échanges de données exigeant une grande fiabilité, le protocole de transport TCP (*Transmission Control Protocol*) est utilisé. Pour ceux qui ne nécessitent pas une telle fiabilité, un protocole de transport plus simple, appelé UDP (*User Datagram Protocol*) fournit un service de bout en bout en mode sans connexion. Avant de décrire les protocoles TCP et UDP, nous allons voir les notions qui leur sont communes.

Le protocole de transport, greffé au-dessus d'IP, est choisi en fonction de la qualité de service souhaitée par l'application. Celle-ci choisira l'un des deux protocoles disponibles : TCP, si elle souhaite une grande fiabilité de l'échange des données, ou UDP si elle ne souhaite pas être ralentie par la gestion des services assurant l'intégrité des données. Les deux diffèrent par bien des aspects mais utilisent des concepts communs, notamment les notions de ports, de sockets et de total de contrôle.

1 Notions utilisées dans les protocoles de transport

Un protocole de transport offre aux applications situées sur la machine de l'utilisateur une interface leur permettant d'utiliser les services offerts par le ou les réseaux physiques sous-jacents. Comme plusieurs applications sont susceptibles d'accéder au réseau, il faut les identifier sans ambiguïté ; on utilise pour cela la notion de *port*. Par ailleurs, les extrémités qui échangent des données sont repérées grâce à la notion de *socket*.

Un protocole de transport souhaite en outre s'assurer que l'en-tête contenant les informations nécessaires à la gestion du protocole n'a pas été altéré au cours de la transmission. Il emploie à cette fin des techniques de redondance appelées *total de contrôle* (ou *checksum*).

1.1 Notion de port

Les identifiants d'application sont indispensables, car plusieurs applications différentes peuvent s'exécuter simultanément sur la même machine. Un protocole de transport doit donc savoir pour qui il travaille ! L'identifiant d'application est appelé *numéro de port* – ou *port* – (ce qui n'a rien à voir avec les ports d'un commutateur). Selon les systèmes d'exploitation, le numéro de port peut correspondre au PID (*Process Identifier*), l'identifiant du processus qui s'exécute sur la machine.

Il existe des milliers de ports utilisables, puisque le numéro de port tient sur 16 bits. Une affectation officielle des ports a été définie par l'IANA (*Internet Assigned Numbers Authority*), afin d'aider à la configuration des réseaux. Parmi ceux-ci, les ports 0 à 1023 sont les ports bien connus ou réservés (*well known ports*), affectés aux processus système ou aux programmes exécutés par les serveurs. Les systèmes d'exploitation diffèrent dans leur affectation pour les identificateurs des autres processus. Ils choisissent *a priori* les grands numéros. Le tableau 7.1 cite quelques numéros de port bien connus.

Tableau 7.1

Quelques ports bien connus

Port	Service	Port	Service
20 et 21	FTP	989 et 990	FTPS
22	SSH		
23	Telnet		
25	SMTP		
53	DNS		
80	Web	443	HTTPS
110	POP3	995	POP3S
143	IMAP	993	IMAPS
161	SNMP		
179	BGP		
520	RIP		



1.2 Interface entre le protocole de transport et l'application

Les applications ayant le choix du protocole de transport (TCP ou UDP), le système d'exploitation doit utiliser un système de nommage qui affecte un identifiant unique, appelé *socket*[1], à tout processus local utilisant les services d'un protocole de transport. Le socket est constitué de la paire :

< *adresse IP locale, numéro de port local* >.

Pour identifier de manière unique l'échange de données avec le processus applicatif distant, le protocole de transport utilise un ensemble de cinq paramètres formé par le nom du protocole utilisé, le socket local et le socket distant :

< *protocole, adresse IP locale, numéro de port local ; adresse IP distante, numéro de port distant* >.

1.3 Total de contrôle ou checksum

Un total de contrôle est une information de redondance permettant de contrôler l'intégrité d'un bloc d'informations défini. Le calcul effectué est en général plus simple que celui décrit au chapitre 2 ; il s'agit, le plus souvent, d'une simple addition sans report des bits du bloc (un OU exclusif), suivie éventuellement d'une complémentation bit à bit du résultat précédent. Dans TCP ou dans UDP, le total de contrôle est un champ de 16 bits incorporé dans l'en-tête. Sa position dépend du protocole de transport utilisé.

2 Protocole TCP (*Transmission Control Protocol*)

TCP comble les carences d'IP lorsque les applications requièrent une grande fiabilité. Ce protocole de transport, lourd et complexe, met en œuvre la détection et la correction d'erreurs, gère le contrôle de flux et négocie les conditions du transfert des données entre les deux extrémités de la connexion.

L'entité gérée par le protocole TCP s'appelle le *segment*. Une fois le segment fabriqué, le module TCP sollicite le module IP pour le service de transmission, par l'intermédiaire de la primitive que nous avons vue au chapitre précédent : *Requête_émission (segment, adresse IP distante)*, dans laquelle les deux paramètres fournis sont le segment à émettre et l'adresse IP de destination. (Dans la pratique, plusieurs autres paramètres sont également fournis, mais nous nous intéressons ici au principe de fonctionnement.) À l'inverse, lorsque le module IP reçoit un datagramme destiné à la machine concernée et que celui-ci transporte un segment TCP, le module IP extrait le segment du datagramme et en signale l'arrivée au module TCP par la primitive : *Indication_réception (segment reçu, adresse IP source)*.

L'interface entre la couche TCP et la couche IP est très simple, celle entre la couche TCP et l'application utilisatrice est beaucoup plus complexe. Nous la détaillerons plus loin, après avoir vu le format du segment TCP et la vie d'une connexion.

1. Nous avons conservé ce terme anglais car aucun équivalent français n'est utilisé.

2.1 Dialogue de bout en bout

À la demande des applications qui ont besoin d'échanger des données de manière fiable, TCP ouvre une connexion et gère un dialogue. TCP n'est implanté que sur les machines des utilisateurs, c'est-à-dire qu'il n'est pas géré par les systèmes intermédiaires (ponts, commutateurs et autres routeurs du réseau). Il est défini dans la RFC 793 et s'occupe de gérer et de fiabiliser les échanges, alors qu'IP est responsable de la traversée des différents réseaux.

TCP permet à deux utilisateurs d'avoir une vision de bout en bout de leurs échanges, quels que soient l'interconnexion de réseaux sous-jacente et le chemin par lequel IP a fait passer les données. Pour apporter la fiabilité dont les utilisateurs ont besoin, TCP gère un *contexte* de l'échange (l'ensemble des paramètres choisis et négociés par les deux utilisateurs pour leur dialogue, ainsi que l'ensemble des paramètres temporels liés à l'état du dialogue). Le contexte est mémorisé dans le module TCP des deux interlocuteurs. Ce protocole fonctionnant en mode connecté, les deux modules TCP doivent être opérationnels simultanément. En outre, il faut que l'une des extrémités ait sollicité l'autre et que cette dernière ait répondu positivement. On parle de la *vie* de la connexion pour décrire tous les événements qui s'y produisent.

2.2 Fonctionnalités de TCP

TCP est capable de détecter les datagrammes perdus ou dupliqués et de les remettre dans l'ordre où ils ont été émis. Ce service repose sur la numérotation et l'acquittement des données et utilise une fenêtre d'anticipation. Remarquons dès à présent que la numérotation des données dans TCP s'effectue octet par octet, alors que les protocoles définis dans le modèle OSI numérotent les unités de données du niveau concerné.

TCP considère les données transportées comme un *flot* non structuré d'octets. Une connexion étant *a priori* bidirectionnelle, les deux flots de données sont traités indépendamment l'un de l'autre. Le flot géré par chaque module concerne les octets de données compris dans une zone délimitée nommée *fenêtre*, dont la taille est définie par un entier de 16 bits, ce qui la limite *a priori* à 65 535 octets. Chaque module TCP annonce la taille de son tampon de réception à l'ouverture de la connexion. Il est convenu que l'émetteur n'envoie pas plus de données que le récepteur ne peut en accepter. La taille de la fenêtre varie en fonction de la nature du réseau et surtout de la bande passante estimée à partir des mesures de performances qui sont effectuées régulièrement (voir section 2.4).

Grâce aux mesures effectuées, différents temporisateurs d'attente maximale d'acquittement de bout en bout sont dimensionnés de manière dynamique, à partir de la connaissance acquise sur le fonctionnement du réseau. En effet, le délai de traversée du réseau change d'une connexion à l'autre ; il peut même changer pendant la vie d'une connexion puisque IP ne garantit rien, pas même l'ordre dans lequel arrivent les données. Par ailleurs, TCP gère un flot de données urgentes, non soumises au contrôle de flux.



Remarque

Par bien des aspects, le protocole de transport TCP ressemble au protocole de liaison de données du modèle OSI, puisque tous les deux gèrent le séquencement des données, le contrôle de flux, la détection et la récupération des erreurs. Il existe toutefois des différences importantes : tout d'abord, les entités de liaison communiquent directement sur le support physique, alors que les entités de transport communiquent à travers un ou plusieurs réseaux interconnectés. De ce fait, l'entité de transport doit spécifier l'adresse du destinataire. En outre, une entité de transport peut gérer un nombre important et variable de connexions, alors que l'entité de liaison n'en gère le plus souvent qu'une seule. Enfin, les données transmises dans l'interconnexion de réseaux peuvent « tourner en rond » ou disparaître pour réapparaître un peu plus tard. Elles peuvent donc être reçues dans un ordre qui n'était pas celui de leur émission, ce qui ne peut pas se produire sur un support physique unique.

2.3 Format du segment TCP

Il faut remarquer qu'il n'y a qu'un seul format de segment TCP, illustré à la figure 7.1, bien que le protocole soit complexe. Le segment contient un en-tête de 20 octets (sauf options) et un champ de données.

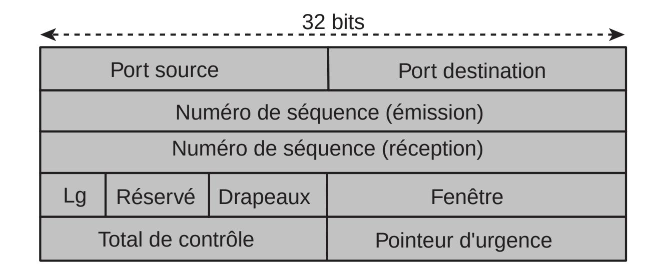


Figure 7.1

Format du segment TCP (en-tête sans option).

Signification des différents champs

- *Port Source* (16 bits). Numéro du port utilisé par l'application en cours sur la machine source.
- *Port Destination* (16 bits). Numéro du port relatif à l'application en cours sur la machine de destination.
- *Numéro d'ordre* (32 bits). La signification de ce numéro est à interpréter selon la valeur du drapeau SYN (*Synchronize*). Lorsque le bit SYN est à 0, le numéro d'ordre est celui du premier octet de données du segment en cours (par rapport à tous les octets du flot de données transportées). Lorsqu'il est à 1, le numéro d'ordre est le *numéro initial*, celui du premier octet du flux de données qui sera transmis (*Initial Sequence Number*). Celui-ci est tiré au sort, plutôt que de commencer systématiquement à 0[2].
- *Numéro d'accusé de réception* (32 bits). Numéro d'ordre du dernier octet reçu par le récepteur (par rapport à tous les octets du flot de données reçues).
- *Longueur en-tête* (4 bits). Il permet de repérer le début des données dans le segment. Ce décalage est essentiel, car il est possible que l'en-tête contienne un champ d'options

2. Le numéro de séquence initial est géré par une horloge interne, propre à la machine. Par exemple, si cette horloge est à 200 MHz, les numéros changent toutes les cinq microsecondes. Comme ce numéro est codé sur 32 bits, il y 4 milliards de numéros. Un même numéro ressort toutes les cinq heures et demie environ !

de taille variable. Un en-tête sans option contient 20 octets, donc le champ longueur contient la valeur 5, l'unité étant le mot de 32 bits (soit 4 octets).

- *Réservé* (6 bits). Champ inutilisé (comme dans tous les formats normalisés, il reste une petite place, prévue en cas d'évolutions à venir ou en cas de bogue à corriger, par exemple).
- *Drapeaux* ou *flags* (6 bits). Ces bits sont à considérer individuellement :
  - *URG (Urgent).* Si ce drapeau est à 1, le segment transporte des données urgentes dont la place est indiquée par le champ Pointeur d'urgence (voir ci-après).
  - *ACK (Acknowledgement).* Si ce drapeau est à 1, le segment transporte un accusé de réception.
  - *PSH (Push).* Si ce drapeau est à 1, le module TCP récepteur ne doit pas attendre que son tampon de réception soit plein pour délivrer les données à l'application. Au contraire, il doit délivrer le segment immédiatement, quel que soit l'état de son tampon (méthode Push).
  - *RST (Reset).* Si ce drapeau est à 1, la connexion est interrompue.
  - *SYN (Synchronize).* Si ce drapeau est à 1, les numéros d'ordre sont synchronisés (il s'agit de l'ouverture de connexion).
  - *FIN (Final).* Si ce drapeau est à 1, la connexion se termine normalement.
- *Fenêtre* (16 bits). Champ permettant de connaître le nombre d'octets que le récepteur est capable de recevoir sans accusé de réception.
- *Total de contrôle* ou *checksum* (16 bits). Le total de contrôle est réalisé en faisant la somme des champs de données et de l'en-tête. Il est calculé par le module TCP émetteur et permet au module TCP récepteur de vérifier l'intégrité du segment reçu.
- *Pointeur d'urgence* (16 bits). Indique le rang à partir duquel l'information est une donnée urgente.
- *Options* (taille variable). Options diverses, les plus fréquentes étant :
  - *MSS* (*Maximum Segment Size*). Elle sert à déterminer la taille maximale du segment que le module TCP accepte de recevoir. Au moment de l'établissement d'une connexion, le module émetteur annonce sa taille de MSS.

Exemple

Pour une application qui s'exécute sur un réseau Ethernet dans un environnement TCP/IP, le paramètre MSS pourra être 1 460 octets, soit 1 500 – taille maximale du champ de données d'une trame Ethernet – moins 40 octets, c'est-à-dire deux en-têtes de 20 octets (la taille normale de l'en-tête du datagramme IP sans option et de celle du segment TCP).

  - *Timestamp* (*estampille temporelle*). Sert à calculer la durée d'un aller et retour (RTT, *Round Trip Time*).
  - *Wscale* (*Window Scale* ou facteur d'échelle). Sert à augmenter la taille de la fenêtre au-delà des 16 bits du champ Fenêtre normal. Si la valeur proposée est *n*, alors la taille maximale de la fenêtre est de 65 535*2^n.
- *Remplissage.* Les options utilisent un nombre quelconque d'octets, or les segments TCP sont toujours alignés sur une taille multiple entier de 4 octets. Si l'en-tête sans option compte 5 mots de 32 bits, les options peuvent avoir une taille quelconque. Si besoin, on remplit l'espace qui suit les options avec des zéros pour aligner la taille du segment à une longueur multiple de 32 bits.
- *Données.* Ce champ transporte les données normales et éventuellement les données urgentes du segment.

2.4 Connexion TCP

TCP est un protocole qui fonctionne en mode client/serveur : l'un des utilisateurs est le serveur offrant des services, l'autre est le client qui utilise les services proposés par le serveur.

Ouverture de la connexion TCP

Le serveur doit être initialisé le premier ; on dit qu'il exécute une *ouverture passive*. Dès qu'il est opérationnel, il attend les demandes des clients, qui peuvent alors faire une *ouverture active*. Pour ouvrir une connexion, les clients doivent connaître le numéro de port de l'application distante. En général, les serveurs utilisent des numéros de ports bien connus, mais il est possible d'implanter une application sur un numéro de port quelconque. Il faut alors prévenir les clients pour qu'ils sachent le numéro de port à utiliser.

Le client ouvre la connexion en envoyant un premier segment, parfois appelé *séquence de synchronisation*. Ce segment contient en particulier le numéro de séquence initial (le numéro du premier octet émis) et le drapeau SYN est mis à 1. Le serveur répond par un acquittement comprenant le numéro du premier octet attendu (celui qu'il a reçu + 1) et son propre numéro de séquence initial (pour référencer les octets de données du serveur vers le client). Dans ce segment de réponse, le serveur a positionné les drapeaux SYN et ACK à 1. Enfin, le client acquitte la réponse du serveur en envoyant un numéro d'acquittement égal au numéro de séquence envoyé par le serveur + 1. Dans ce troisième message, seul le drapeau ACK est mis à 1.

L'ensemble des trois segments correspond à l'ouverture de la connexion, illustrée à la figure 7.2. Ce mécanisme d'ouverture est appelé *three-way-handshake* (établissement en trois phases). Après la dernière phase, le transfert des données peut commencer.

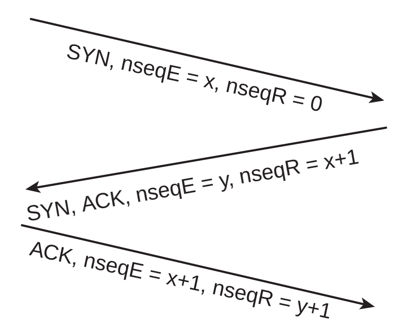


Figure 7.2

Ouverture d'une connexion TCP.

Fermeture de la connexion TCP

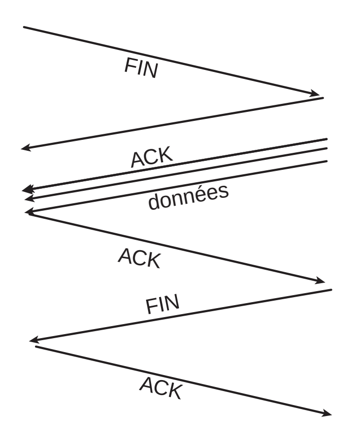
Une connexion TCP étant bidirectionnelle, les flots de données circulent dans les deux sens et chaque sens est géré par une extrémité. Les flots doivent donc être arrêtés indépendamment l'un de l'autre. De ce fait, si trois segments sont échangés pour établir une connexion, il en faut quatre pour qu'elle s'achève correctement, comme le montre la figure 7.3.

Pour indiquer qu'il a terminé l'envoi des données, un des modules TCP envoie un segment avec le drapeau FIN mis à 1. Ce segment doit être acquitté par le module distant. La connexion n'est vraiment fermée que lorsque les deux modules ont procédé à cet échange. La fermeture définitive n'intervient qu'après expiration d'un temporisateur.

Remarquons qu'une connexion peut être brutalement fermée à la suite d'un incident (par exemple lorsque l'utilisateur ferme inopinément son application). Le module TCP qui arrête brutalement la connexion émet un segment avec le drapeau RST mis à 1. Ce segment contient éventuellement les derniers octets en attente et aucun acquittement n'en est attendu. Le module TCP distant qui reçoit un segment avec le bit RST à 1 transmet les éventuelles dernières données à l'application et lui signale la fin anormale de la connexion.

Figure 7.3

Fermeture d'une connexion TCP.



Automate de gestion d'une connexion TCP

On voit que l'ouverture et la fermeture d'une connexion dépendent de nombreux événements et peuvent conduire à beaucoup de situations différentes. Une représentation commode des étapes de l'établissement et de la libération d'une connexion TCP se fait en utilisant un automate d'états finis à onze états (voir tableau 7.2). Dans chaque état, seuls certains événements sont autorisés et une action précise est entreprise. Dans le cas où l'événement n'est pas autorisé, on renvoie un signal d'erreur. La figure 7.4 donne les transitions d'états de l'automate d'une connexion TCP.

Tableau 7.2

Les états d'une connexion TCP

État	Description
CLOSED / FERMÉ	Aucune connexion active ou en attente
LISTEN / ÉCOUTE	Attente d'appel entrant par le serveur
SYN RCVD / SYN RECU	Attente d'ACK après l'arrivée de requête de connexion
SYN SENT / SYN ENVOYÉ	Ouverture de connexion déjà commencée par l'application
ESTABLISHED / ÉTABLIE	État normal de transfert de données
FIN WAIT1 / FIN ATTENTE 1	L'application a terminé
FIN WAIT2 / FIN ATTENTE 2	L'autre extrémité est d'accord pour libérer la connexion
TIME WAIT /TEMPORISATION	Attente de la fin d'émission des segments
CLOSING / FERMETURE	Tentative de fermeture de connexion
CLOSE WAIT / ATTENTE DE FERMETURE	L'autre extrémité a initialisé une fermeture de connexion
LASTACK / DERNIER ACK	Attente de la fin d'émission des segments



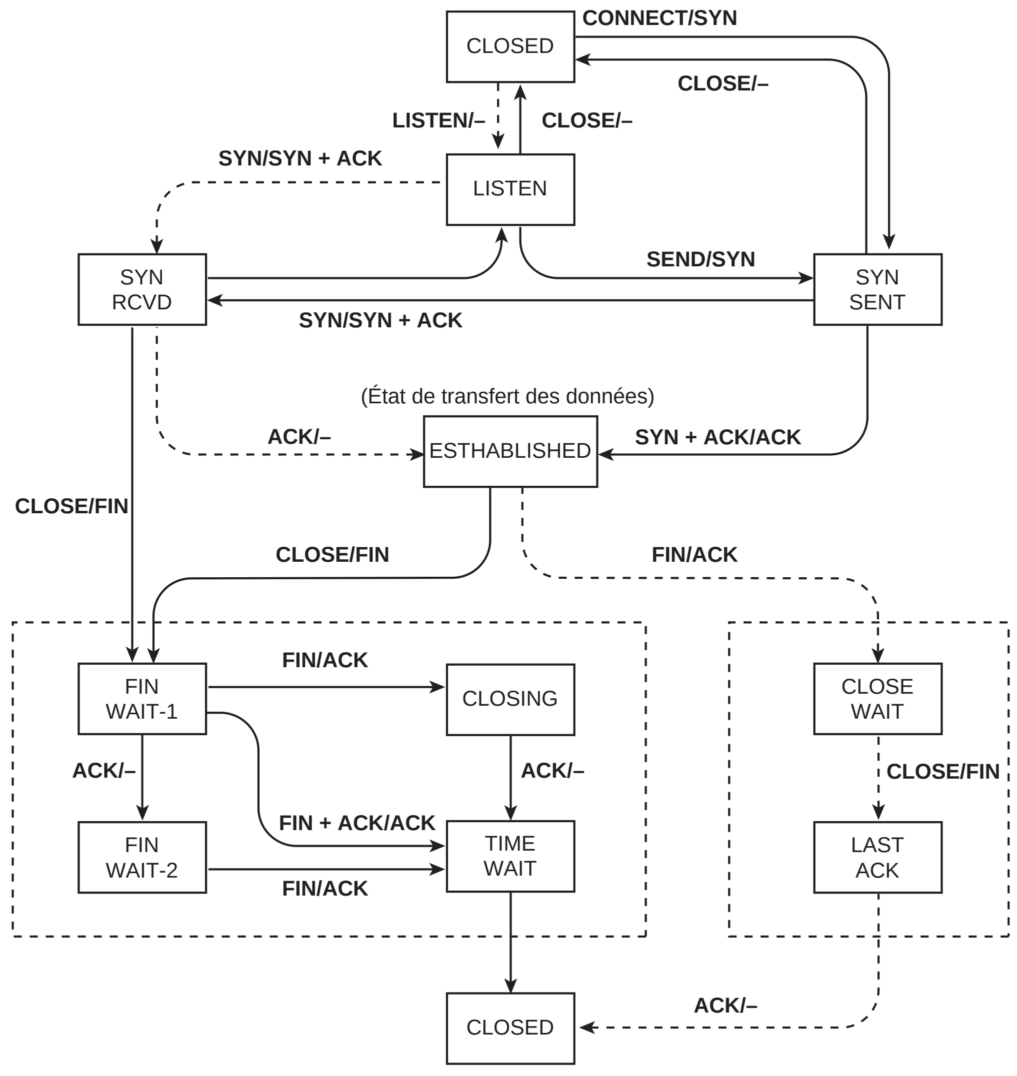


Figure 7.4

Diagramme de transitions d'états dans TCP.

Exemple

Pour obtenir des informations, en particulier sur les connexions TCP gérées par une machine, il faut taper la commande : `netstat s p tcp`.

Le tableau 7.3 montre les différents états des connexions TCP de la machine.

Tableau 7.3

Connexions TCP et leurs états

Protocole	Adresse locale	Adresse distante	État
TCP	ma_machine:1039	Pluton… : netbios	*ESTABLISHED*
TCP	ma_machine:1040	Uranus… : 993	*CLOSE-WAIT*
TCP	ma_machine:1026	207.46.19.30 : http	*ESTABLISHED*
TCP	ma_machine:1051	193.51.224.15 : http	*ESTABLISHED*
TCP	ma_machine:1055	Mercure… : https	*TIME-WAIT*

Transfert des données

Un des concepts les plus importants et les plus complexes de TCP est lié à sa façon de gérer les temporisations et les retransmissions. Comme d'autres protocoles fiables, TCP suppose

que le destinataire émet des accusés de réception chaque fois qu'il reçoit de nouvelles données valides. Il arme une temporisation chaque fois qu'il émet un segment, puis il attend de recevoir l'accusé de réception correspondant. Si le délai d'attente d'acquittement est atteint avant que les données du segment ne soient acquittées (on dit que la temporisation expire), TCP suppose que ce segment a été détruit ou perdu et le retransmet.

TCP est destiné à être utilisé dans un réseau quelconque, un petit réseau local ou dans Internet. Un segment qui transite d'un ordinateur à l'autre peut traverser un seul réseau à faible temps de transit ou traverser un ensemble de réseaux et de routeurs intermédiaires. Il est impossible, *a priori*, de connaître la rapidité avec laquelle les accusés de réception seront reçus par la source. De plus, le délai dépend du trafic ; il peut ainsi subir des variations considérables d'un moment à l'autre. Le temps d'aller et retour ou RTT (*Round Trip Time*) mesure le temps mis pour traverser le réseau. TCP doit donc prendre en compte à la fois les variations des délais de retransmission pour les différentes destinations et la grande dispersion des délais pour une destination donnée.

Exemple

L'utilitaire `ping` permet de savoir si une machine est opérationnelle et fournit une estimation du délai aller et retour. Dans l'exemple qui suit, la première machine est dans le même réseau local que celle de l'émetteur (adresse privée), la seconde est en Australie !

```
ping 192.168.0.2
délai approximatif des boucles en millisecondes
minimum = 3 ms, maximum = 11 ms, moyenne = 6 ms
ping 203.50.4.178
délai approximatif des boucles en millisecondes
minimum = 355 ms, maximum = 361 ms, moyenne = 358 ms
```

Algorithme de Jacobson Le module TCP surveille le comportement de chaque connexion et en déduit des valeurs de temporisation raisonnables. Il s'adapte aux variations de délais en modifiant les valeurs de temporisation. Le choix de la durée d'une temporisation est difficile, car la mesure du temps aller et retour dans le réseau est complexe. En outre, même si cette valeur est connue, il est difficile de fixer une temporisation : trop courte, elle provoque d'inutiles retransmissions ; trop longue, elle fait chuter les performances. L'*algorithme de Jacobson* est un algorithme dynamique qui ajuste la valeur de cette temporisation en fonction de mesures prises à intervalles réguliers dans le réseau.

On recueille des informations – comme la date à laquelle un segment TCP est émis et celle à laquelle l'accusé de réception correspondant lui parvient – pour calculer le RTT, qui est actualisé à chaque transmission. Le module TCP retient une estimation pondérée de RTT qui s'appuie à la fois sur le présent (*Nouveau_RTT*) et le passé (*RTT_estimé*). Le calcul de cette estimation tient compte d'un facteur de pondération a compris entre 0 et 1, donné par la formule :

$$RTT_estimé = a*RTT_estimé + (1 - a)*Nouveau_RTT.$$

Le choix d'une valeur de a proche de 1 rend la valeur estimée de RTT insensible aux variations brèves, contrairement au choix d'une valeur de a proche de 0. Le plus souvent, on prend $a = 7/8$. TCP calcule finalement une valeur de temporisation à partir de la valeur *RTT_estimé* :

$$Temporisation = b*RTT_estimé \text{ (avec } b > 1).$$

La valeur de temporisation doit être proche de la valeur *RTT_estimé* pour détecter les pertes de segment aussi vite que possible : si b vaut 1, tout retard provoque des retransmissions inutiles ; la valeur $b = 2$ n'est pas optimale lorsque la variance change. Jacobson proposa un algorithme qui rend b variable, par un calcul de probabilité donnant la valeur du délai de retour d'un accusé de réception : plus la variance croît et plus b est élevé et réciproquement. L'algorithme demande de conserver la trace d'une autre variable,



l'écart D. Chaque fois qu'un accusé de réception arrive, on calcule la valeur absolue de l'écart entre la valeur estimée et la valeur observée : $|RTT_estimé - Nouveau_RTT|$ et D est alors donné par la formule :

$$D = (1 - a)*|RTT_estimé - Nouveau_RTT|.$$

Dans cette formule, a peut avoir la même valeur que celle utilisée pour pondérer RTT car la valeur de D qui en résulte est suffisamment proche du résultat donné par le calcul de probabilité. On peut ainsi utiliser des opérations simples pour calculer D, afin d'obtenir rapidement le résultat du calcul. La plupart des implantations utilisent cet algorithme et prennent comme valeur de temporisation :

$$Temporisation = RTT_estimé + 4*D.$$

Algorithme de Karn Un autre problème a été identifié avec le mode d'acquittement de TCP : l'*ambiguïté des accusés de réception*. En effet, TCP utilise une technique d'acquittement cumulatif des octets reçus, dans laquelle l'accusé de réception concerne les données elles-mêmes et non pas le segment qui a servi à les acheminer. L'exemple donné ci-après illustre ce phénomène.

L'algorithme de Karn lève l'ambiguïté des accusés de réception. Lors du calcul de la valeur estimée du RTT, on ignore les mesures qui correspondent à des segments retransmis, mais on utilise une stratégie d'augmentation des temporisations (*timer back off strategy*). On conserve la valeur de temporisation obtenue, tant qu'une nouvelle mesure n'a pas été faite. Cela revient donc à calculer une valeur de temporisation initiale à l'aide de la formule précédente. À chaque expiration de la temporisation, TCP augmente la valeur de la temporisation (celle-ci est toutefois bornée pour éviter qu'elle devienne trop longue).

$$Nouvelle\ temporisation = g*Temporisation\ (généralement\ avec\ g = 2).$$

Exemple d'ambiguïté des acquittements de *TCP*

TCP fabrique un segment, le donne à IP qui l'encapsule dans un datagramme puis l'envoie dans le réseau. Si la temporisation expire, TCP retransmet le même segment qui sera donc encapsulé dans un autre datagramme. Comme les deux segments transportent exactement les mêmes données, le module TCP émetteur n'a aucun moyen de savoir, à réception d'un acquittement, si celui-ci correspond au segment initial ou à celui qui a été retransmis : les accusés de TCP sont donc ambigus.
En effet, associer l'accusé de réception à la transmission initiale peut provoquer une forte augmentation du RTT en cas de perte de datagrammes IP : si un accusé de réception arrive après une ou plusieurs retransmissions et que TCP se contente de mesurer le RTT à partir du segment initial, il calcule un nouveau RTT à partir d'un échantillon particulièrement grand, donc il va augmenter sensiblement la temporisation du segment suivant. Si un accusé de réception arrive après une ou plusieurs retransmissions, le RTT suivant sera encore plus grand et ainsi de suite !

Autres temporisateurs TCP utilise trois autres temporisateurs : *de persistance*, *de limitation d'attente* et *de fermeture de connexion*. Le premier permet d'éviter la situation de blocage mutuel suivante : le récepteur envoie un accusé de réception avec une fenêtre nulle pour demander à l'émetteur de patienter. Un peu plus tard, le récepteur met la fenêtre à jour mais le segment est perdu : émetteur et récepteur s'attendent mutuellement. Quand le temporisateur de persistance expire, l'émetteur envoie un message « sonde » (*probe*) au récepteur. La réponse à ce message donne la taille de la fenêtre actuelle. Si elle est toujours nulle, l'émetteur réarme son temporisateur de persistance ; sinon, il peut envoyer des données.

Quand une connexion est inactive depuis un certain temps, le temporisateur de limitation d'attente expire et permet à une extrémité de vérifier si l'autre est toujours présente. On ferme la connexion s'il n'y a pas de réponse. Ce mécanisme n'est pas implanté partout car il augmente la charge du réseau et peut conduire à fermer une connexion active à cause d'une coupure passagère. D'un autre côté, il peut être intéressant de fermer une connexion inactive pour récupérer des ressources : pour toute connexion gérée, le module TCP en mémorise tous les paramètres et donc monopolise des ressources pour la gestion de cette connexion.

Le dernier temporisateur gère l'état *TIME WAIT* pendant la fermeture de connexion. Il vaut deux fois la durée de vie maximale d'un segment, ce qui assure que tous les segments d'une connexion ont disparu quand on la ferme.

Algorithmes de Clark et Nagle Un autre mode de fonctionnement, appelé *syndrome de la fenêtre stupide* (*silly window syndrom*), peut contribuer à faire s'effondrer les performances de TCP. Ce cas se rencontre lorsque l'application donne de grands blocs de données à l'entité TCP émettrice, alors que l'entité réceptrice traite les données octet par octet (une application interactive par exemple). Au départ, le tampon mémoire du récepteur est plein et l'émetteur le sait car il a reçu une indication de fenêtre de taille 0. L'application lit un caractère du flux ; le module TCP du récepteur envoie une indication d'actualisation de fenêtre pour qu'on lui envoie l'octet suivant. L'émetteur se croit obligé d'envoyer un octet et le tampon est à nouveau plein. Le récepteur acquitte le segment d'un octet mais positionne la fenêtre à 0 et ainsi de suite…

La solution de Clark consiste à empêcher le récepteur d'envoyer une indication d'actualisation de fenêtre pour un seul octet. On oblige le récepteur à attendre qu'il y ait suffisamment d'espace disponible avant de l'annoncer. En pratique, il n'enverra pas d'indication d'actualisation de fenêtre tant qu'il n'a pas atteint la taille maximale du segment. Le récepteur a obtenu cette valeur à l'établissement de la connexion, ou lorsque son tampon mémoire est à moitié vide (on prend généralement le minimum de ces deux valeurs). L'émetteur peut aider en n'envoyant pas de petits segments. Pour cela, il s'appuie sur les estimations qu'il établit à partir des indications d'actualisation de fenêtre déjà reçues. L'algorithme de Nagle propose d'envoyer le premier octet seul et d'accumuler les autres octets dans un tampon, tant que le premier octet n'est pas acquitté, puis d'envoyer dans un seul segment toutes les données accumulées et ainsi de suite. Cet algorithme est largement employé dans les implantations TCP mais dans certains cas, il est préférable de le désactiver. Les algorithmes de Clark et de Nagle sont complémentaires et utilisables simultanément.

L'objectif est que l'émetteur n'envoie pas de petits segments et que le récepteur n'en réclame pas en émettant des indications d'actualisation de fenêtre avec des valeurs trop faibles. En effet, le TCP récepteur, tout comme le TCP émetteur, peut accumuler les données. Pour cela, il bloque les primitives de lecture de l'application, tant qu'il n'a pas de données en nombre suffisant à fournir. Cette façon d'opérer diminue le nombre d'appels à TCP et diminue la surcharge globale (elle augmente légèrement le temps de réponse mais ce n'est pas très gênant pour les applications interactives).

Contrôle de congestion par TCP TCP manipule dynamiquement la taille de la fenêtre, pour ne pas injecter de nouveau segment tant qu'il en reste un ancien dans le réseau. On part de l'hypothèse que l'expiration d'un temporisateur sur Internet ne peut provenir que de la congestion d'une partie du réseau, la perte due à une erreur de transmission étant considérée comme un événement rare. TCP surveille donc en permanence les temporisateurs pour détecter toute congestion potentielle. Les capacités d'émission et de réception constituent deux sources possibles de problèmes et doivent être traitées séparément. Pour

cela, chaque émetteur gère deux fenêtres : celle qui est accordée par le récepteur et la *fenêtre de congestion*. Le nombre d'octets envoyés est égal au minimum des deux fenêtres ; celle qui est réellement utilisée correspond au minimum de ce qui convient à l'émetteur et au récepteur.

À l'initialisation de la connexion, l'émetteur prend une fenêtre de congestion correspondant à la taille maximale du segment utilisé et envoie un segment de taille maximale. Si le segment est acquitté avant expiration d'une temporisation, il augmente la taille de la fenêtre de congestion (la nouvelle fenêtre est de taille double par rapport à la précédente), puis l'émetteur envoie deux segments. À chaque accusé de réception de segment, il augmente la fenêtre de congestion en incrémentant d'une unité la taille maximale du segment.

Quand la fenêtre de congestion atteint *n* segments et si tous les segments ont été acquittés dans les temps impartis, l'émetteur augmente la taille de la fenêtre de congestion du nombre d'octets correspondant aux *n* segments. Ainsi, pour chaque émission couronnée de succès, la taille de la fenêtre de congestion va-t-elle doubler : elle croît exponentiellement, jusqu'à expiration de la temporisation ou si on atteint la taille maximale de la fenêtre de réception.

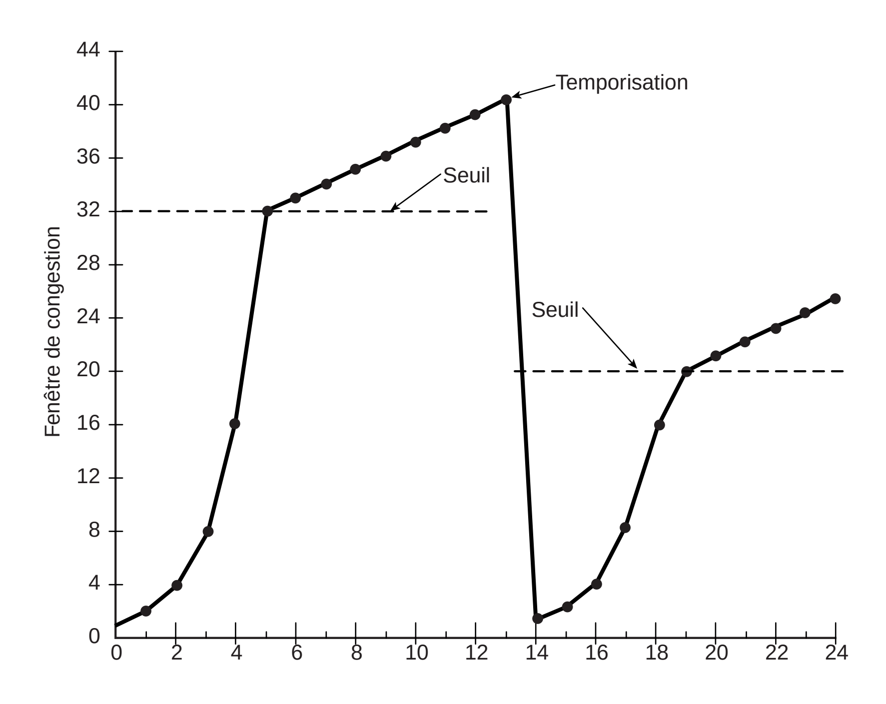
Exemple

Si on a envoyé des segments de 1 024, 2 048, 4 096 octets correctement mais qu'un segment de 8 192 octets provoque une expiration de temporisation, on évitera la congestion en dimensionnant la fenêtre à 4 096 octets et on n'enverra pas de segments de taille supérieure à cette valeur, quelle que soit la taille de la fenêtre de réception. Cet algorithme est appelé *algorithme de démarrage lent* (qui n'est en fait pas lent du tout puisqu'il a une croissance exponentielle !).

Un troisième paramètre, le seuil d'évitement de congestion (*threshold*), a pour valeur initiale 64 Ko. À expiration de la temporisation, le seuil est pris égal à la moitié de la fenêtre de congestion courante, et TCP réinitialise la fenêtre de congestion à la taille maximale de segment. Le démarrage lent est utilisé pour tester les possibilités d'absorption du récepteur ; on arrête la croissance exponentielle lorsque le seuil d'évitement de congestion est atteint. En cas de transmission avec succès, on augmente linéairement la taille de la fenêtre de congestion (c'est-à-dire qu'on augmente d'un segment la taille de la fenêtre au lieu d'un segment par segment acquitté), comme le montre la figure 7.5.

Figure 7.5

Évolution de la fenêtre de congestion.



2.5 INTERFACE ENTRE TCP ET L'APPLICATION

Les processus applicatifs communiquent en utilisant des sockets TCP. La programmation d'une application client/serveur se fait donc en manipulant les sockets.

Côté serveur, il faut d'abord créer le socket et le paramétrer (primitive *Bind*), en lui associant le numéro de port correspondant. Il faut ensuite le placer dans un état d'attente du client (primitive *Listen*).

Côté client, il faut créer le socket et établir la connexion (primitive *Connect*) qui sera acceptée par le serveur (primitive *Accept*). Le transfert des données pourra commencer, en utilisant des primitives *Read* et *Write* (ou *Sendto* et *Receivefrom*).

La fermeture est liée à l'utilisation de la primitive *Close* côté client (fermeture active) ou côté serveur (fermeture passive).

3 Protocole UDP (*User Datagram Protocol*)

UDP est un protocole de transport sans connexion qui permet l'émission de messages sans l'établissement préalable d'une connexion. C'est un protocole non fiable, beaucoup plus simple que TCP, car il n'ajoute aucune valeur ajoutée par rapport aux services offerts par IP. L'utilisateur n'est donc pas assuré de l'arrivée des données dans l'ordre où il les a émises, pas plus qu'il ne peut être sûr que certaines données ne seront ni perdues, ni dupliquées, puisqu'UDP ne dispose pas des mécanismes de contrôle pour vérifier tout cela. De ce fait, il n'introduit pas de délais supplémentaires dans la transmission des données entre l'émetteur et le récepteur. C'est la raison pour laquelle il est utilisé par les applications qui ne recherchent pas une grande fiabilité des données ou qui ne veulent pas assumer la lourdeur de gestion des mécanismes mis en jeu dans le mode connecté.

3.1 SERVICE MINIMAL

UDP est efficace pour le transfert des serveurs vers les clients avec des débits élevés, en délivrant les données sous forme de datagrammes de petite taille et sans accusé de réception. Ce type de service est utile pour les applications en temps réel, telles que les émissions en flux continu d'informations audio et vidéo. En effet, pour celles-ci, la perte d'une partie des données n'a pas grande importance. Les jeux en réseau, le *streaming* (procédé permettant de lire des fichiers audio ou vidéo avant même que celui-ci soit totalement téléchargé, grâce à une mise en mémoire tampon) utilisent aussi UDP. D'autres applications de type questions-réponses, comptant de petites quantités de données, peuvent également utiliser UDP. De ce fait, l'erreur ou la perte d'un datagramme sont gérées directement par l'application elle-même, le plus souvent à l'aide d'un mécanisme de temporisation. Au-dessus d'UDP, on trouve en particulier : le service d'annuaire DNS (*Domain Name System*), la transmission des informations de gestion de réseaux SNMP (*Simple Network Management Protocol*) ou d'informations de routage RIP (*Routing Information Protocol*).

Notons que nous avons déjà vu (chapitre 6, exercice 15), un exemple utilisant le protocole UDP avec la commande *traceroute* sous Unix : celle-ci génère des datagrammes UDP, placés dans des datagrammes IP avec des durées de vie délibérément trop courtes.



3.2 Format du datagramme UDP

Les messages UDP sont généralement appelés *datagrammes UDP*. Ils contiennent deux parties, un en-tête et des données encapsulées dans les datagrammes IP, comme les segments TCP. Le format est illustré dans la figure 7.6.

Figure 7.6

Format du datagramme UDP.

32 bits	
Port source	Port destination
Longueur	Total de contrôle

L'en-tête très simple compte quatre champs :

- *Port source* (16 bits). Il s'agit du numéro de port correspondant à l'application émettrice du paquet. Ce champ représente une adresse de réponse pour le destinataire.
- *Port destination* (16 bits). Contient le port correspondant à l'application de la machine à laquelle on s'adresse. Les ports source et destination ont évidemment la même signification que pour TCP.
- *Longueur* (16 bits). Précise la longueur totale du datagramme UDP, exprimée en octets. La longueur maximale des données transportées dans le datagramme UDP est de : $2^{16} – 4*16$, soit 65 472 octets.
- *Total de contrôle* ou *checksum* (16 bits). Bloc de contrôle d'erreur destiné à contrôler l'intégrité de l'en-tête du datagramme UDP, comme dans TCP.

3.3 Interface entre UDP et l'application

Les processus applicatifs utilisent des sockets UDP. Leur manipulation est très simple puisque le protocole n'est pas en mode connecté : il n'y a pas de procédure de connexion et donc pas de fermeture non plus. Comme pour TCP, du côté du serveur, il faut d'abord créer le socket et le paramétrer par la primitive *Bind*, en lui associant le numéro de port correspondant. Puis il faut le placer dans un état d'attente des données du client (primitive *Listen*). Côté client, il faut créer le socket. Le transfert des données peut commencer directement en utilisant des primitives *Read* et *Write* (ou *Sendto* et *Receivefrom*).

Résumé

Deux protocoles de transport sont utilisés dans l'architecture TCP/IP. Le premier, TCP, est un protocole complet, destiné à pallier toutes les défaillances de l'interconnexion de réseaux. C'est un protocole en mode connecté qui met en œuvre une détection et une correction d'erreurs, un contrôle de séquence, de flux et de congestion. Il est de ce fait complexe et lourd à gérer. TCP est indispensable pour toutes les applications qui transfèrent de grandes quantités d'informations et qui ont besoin de fiabilité dans les échanges de données.

UDP, lui, utilise le protocole IP pour acheminer un message d'un ordinateur à un autre, sans aucune valeur ajoutée (pas de connexion, pas de contrôle d'erreur, de contrôle de flux ni de contrôle de séquence) par rapport aux services rendus par IP. Il convient aux applications de type requête/réponse simples ou ayant des contraintes temporelles fortes.

Problèmes et exercices

Exercice 1 Principes et intérêt de TCP

Énoncé

a Sachant qu'un segment TCP contient 20 octets d'en-tête, qu'il est transporté dans un datagramme IP contenant lui aussi 20 octets d'en-tête, déterminez le débit utile maximal d'une application utilisant TCP/IP sur un réseau Ethernet à 10 Mbit/s.

b Quel intérêt y a-t-il pour un protocole comme TCP à ne posséder qu'un seul format d'en-tête ?

c Quel est l'intérêt du fonctionnement en mode connecté pour le protocole TCP ?

d La fragmentation et le réassemblage étant pris en charge par IP, pourquoi TCP se préoccupe-t-il de l'ordre d'arrivée des datagrammes ?

e Un module TCP peut-il gérer plusieurs connexions simultanément ? Si oui, quel en serait l'intérêt ? Comment pourrait-on distinguer ces connexions ?

Solution

a Nous avons vu que le débit maximal sur Ethernet était 9,82 Mbit/s si le débit réel était de 10 Mbit/s (exercice 7 du chapitre 5). Le calcul était fait en supposant que les 1 500 octets de la trame étaient des octets utiles. Si le champ de données de la trame Ethernet transporte un datagramme IP avec un segment TCP encapsulé, il y a (sauf options) 20 + 20 = 40 octets d'en-tête donc seulement, soit 1 460 octets de données utiles. Le débit maximal est donc 10*(1460/1528) = 9,55 Mbit/s.

b Tous les segments TCP ont le même format, qu'il s'agisse de la demande d'ouverture de connexion, d'un segment de transfert de données ou d'une fermeture de connexion : le traitement est donc toujours le même, il peut être optimisé pour une meilleure efficacité.

c L'intérêt est de disposer d'un contexte, mémorisé chez l'émetteur comme chez le destinataire (protocole de bout en bout), dans lequel sont conservés tous les paramètres fixes et variables de la connexion : cela permet de suivre l'évolution de la connexion et d'adapter au mieux les délais pour la mise en œuvre des fonctions de contrôle d'erreur, de contrôle de flux, de séquencement et de congestion.

d TCP reçoit les données extraites des datagrammes IP et les réordonne (seulement dans le cas où tous les fragments sont arrivés), l'ordre des datagrammes IP n'étant pas géré par IP (les datagrammes n'ont pas de numéro de séquence…). C'est pourquoi TCP doit assurer ce service en numérotant les octets du flot de données qu'il a mis dans le segment.

e Oui, bien sûr ! Et même plusieurs centaines simultanément… Par exemple, une application simple comme la navigation sur le Net ouvre (sans que l'utilisateur le sache…) des dizaines de connexions : chaque objet multimédia dans la page consultée correspond à une connexion ; chaque clic de souris fait ouvrir une nouvelle connexion… Une connexion est identifiée par deux sockets (numéro de port local, adresse IP locale ; numéro de port distant, adresse IP distante). Même si l'adresse IP locale est la même, le numéro de port change : il correspond au processus que crée le système d'exploitation de la machine local et les numéros distants peuvent varier. Il ne peut donc jamais y avoir confusion entre deux connexions, même si les adresses IP sont les mêmes, puisque les processus ont des identificateurs différents.

Exercice 2 Identification d'une connexion TCP

Énoncé

Soit une connexion TCP identifiée par son quadruplet :

< adresse IP 123.45.67.89, port 12006, adresse IP 12.34.56.78, port 80 >.
À quoi correspond cette connexion ? Traverse-t-elle un ou plusieurs routeurs ?

Solution

La machine qui a ouvert la connexion est un client (grand numéro de port) qui s'est connecté à un serveur Web (port 80). Les deux machines ont des adresses IP de classe A et appartiennent à des réseaux différents. La connexion traverse donc au moins un routeur.

Exercice 3 Identification de plusieurs connexions TCP

Énoncé

Soit deux réseaux (notés 1 et 2) distants l'un de l'autre et interconnectés par Internet, possédant chacun un routeur (*R1* et *R2*). L'architecture de protocoles utilisée est TCP/IP. Le poste *PC1* du premier réseau communique avec le poste *PC2* du second réseau qui est un serveur offrant deux services : Web et FTP.

a Le logiciel TCP est-il implémenté au niveau du routeur *R1* ? Du routeur *R2* ? Des deux routeurs ?

b *PC1* a déjà une connexion TCP établie avec *PC2* pour le service Web. Peut-il établir une seconde connexion pour le service FTP ? Si oui, comment TCP différencie-t-il les deux connexions ?

c *PC1* a terminé le téléchargement et fermé sa connexion avec le service FTP. La connexion avec le service Web est brutalement interrompue et *PC1* en démarre une nouvelle (toujours avec le même serveur *PC2*). Est-il possible que des segments de la première connexion interfèrent avec ceux de la seconde ?

Solution

a Le logiciel TCP n'existe que dans les postes des utilisateurs (clients ou serveurs). Les routeurs ont la couche IP comme couche de niveau supérieur.

b TCP a la capacité de gérer plusieurs connexions simultanément. *PC1* peut donc avoir plusieurs connexions avec *PC2*. Ces connexions diffèrent par le numéro de port local et par le numéro de port distant, donc pas de confusion possible. Les deux sockets valent respectivement :

< adresse *IP-PC1*, port *x*, adresse *IP-PC2*, port 80 > et < adresse *IP-PC1*, port *y*, adresse *IP-PC2*, port 21 >

c La nouvelle connexion avec le service Web utilise des numéros de séquence pour les octets du flot de données échangées qui sont différents de la connexion précédente, puisque le numéro de séquence initial est tiré au sort pour la nouvelle connexion. Il n'y a donc aucun risque que des segments interfèrent.

Exercice 4 État d'une connexion TCP

Énoncé

En utilisant la commande *netstat*, vous constatez que vous avez une connexion TCP ouverte avec la machine 213.33.44.55, port 1863 dans l'état *ESTABLISHED*. Vous envoyez un *ping* à cette machine et celle-ci ne répond pas. Vous pensez donc que la machine est tombée en panne. Pourtant, vous constatez, après vérification, que votre application est toujours opérationnelle. Expliquez la situation.

Solution L'application est toujours opérationnelle (il s'agit ici de MSN). La machine qui l'héberge ou un routeur intermédiaire situé devant cette machine a filtré votre requête *ICMP Echo Request* pour des raisons de sécurité.

Remarque

Le filtrage pour raison de sécurité est vital pour un serveur. Il est dommage que, de ce fait, la commande *ping* n'ait plus de sens ! Une machine qui ne répond pas n'est pas obligatoirement hors service.

EXERCICE 5 TRAITEMENT D'UN SEGMENT TCP

Énoncé On suppose qu'une connexion TCP est ouverte entre deux utilisateurs *A* et *B*. Comment sont traités les segments dans les deux cas suivants :

a L'émetteur et le récepteur sont connectés au même réseau de type TCP/IP ?

b L'émetteur et le récepteur appartiennent à deux réseaux distincts utilisant TCP/IP, interconnectés grâce à un routeur IP ?

Solution

a L'application émettrice (par exemple sur la machine *A*, port *x*) demande à TCP l'ouverture d'une connexion avec l'application de la machine *B*, port *y*. TCP fabrique donc un segment d'ouverture de connexion avec (port *x*, port *y*, SYN = 1), placé dans un datagramme IP avec *A* comme adresse *IP* émetteur et *B* comme adresse *IP* destinataire.

b Que les deux machines soient dans le même réseau ou non ne change rien au fonctionnement de TCP. Seul le traitement fait à l'interface entre IP et les couches inférieures change.

Dans le premier cas, le datagramme est encapsulé dans une trame du réseau local avec *A* comme adresse MAC émetteur et *B* comme adresse MAC destinataire. Dans le deuxième cas, le datagramme est encapsulé une première fois dans une trame du réseau local de *A* avec *A* comme adresse MAC émetteur et RA (routeur côté A, c'est-à-dire le routeur qui gère la sortie du réseau) comme adresse MAC destinataire. Il est encapsulé une seconde fois dans une trame du réseau local de *B* avec RB (routeur côté B, qui gère l'entrée dans le réseau) comme adresse MAC émetteur et *B* comme adresse MAC destinataire.

Lorsque la machine *B* reçoit le datagramme (quelle que soit la trame qui l'a encapsulé), elle en extrait le segment et avertit le module TCP de l'arrivée d'informations correspondant à un échange provenant d'une machine d'adresse IP *A*. Le module TCP analyse le segment et prévient l'application identifiée par le port *y* de la demande d'ouverture de connexion.

EXERCICE 6 STATISTIQUES DE CONNEXIONS TCP

Énoncé En utilisant la commande *netstat*, on peut obtenir des statistiques sur l'activité TCP d'une machine. Commentez l'affichage obtenu.

```
Ouvertures actives= 4038
Ouvertures passives= 1645
Tentatives de connexion non réussies= 5
Connexions réinitialisées= 166
Connexions courantes= 7
Segments reçus = 112432
Segments envoyés= 107838
Segments retransmis= 102
```



Solution

La machine dispose de 7 connexions TCP actives. Sur l'ensemble des connexions gérées, le trafic est à peu près symétrique (autant de segments reçus que de segments émis…), sous réserve que la taille des données soit équivalente. La qualité de la transmission est bonne puisqu'il y a 102 retransmissions pour 107 736 segments envoyés (107 838 = 107 736 + 102) ; le taux d'erreur sur les segments est de $9{,}5*10^{-4}$. Si l'on est dans un réseau Ethernet avec des segments de 1 460 octets utiles pour des trames de 1 500 octets, cela fait un taux d'erreur d'environ $2*10^{-8}$.

EXERCICE 7 STATISTIQUES DÉTAILLÉES DE CONNEXIONS TCP

Énoncé

Voici un autre affichage, plus détaillé, des statistiques d'utilisation de TCP sur une deuxième machine :

```
369091 packets sent
352383 data packets (489293303 bytes)
8 data packets (49 bytes) retransmitted
15876 ack only packets (15016 delayed)
3 URG only packets
13 window probe packets
5 window update packets
803 control packets
224123 packets received
204197 acks (for 487292645 bytes)
315 duplicate acks
0 acks for unsent data
17100 packets (2306201 bytes) received in sequence
5 completely duplicate packets (17 bytes)
2 packets with some duplicated data (8 bytes)
203 out of order packets
0 packets of data after window
0 window probe
1555 window update packets
85 packets received after close
0 discarded for bad checksums
0 discarded for bad header offset fields
0 discarded because packet too short
356 connections requests
92 connections accepts
5 embryonic connections dropped
176379 segments updated rtt (of 176768 attempts)
21 retransmit timeouts
0 connections dropped by retransmit timeout
3 persist timeouts
3132 keepalive timeouts
0 keepalive probes sent
0 connections dropped by keepalive
```

Énoncé (*suite*)

Commentez les lignes ci-dessus, en répondant aux questions suivantes :

a Quelle est la taille moyenne des segments TCP envoyés ? Quelle est la proportion d'acquittements ?

b Quelle est la cause la plus vraisemblable des pertes ?

c À quoi servent les paquets `window probe` et `window update` ?

☞

Énoncé (suite)

d `Que contiennent les paquets de contrôle ?`

e Quelle proportion de paquets transporte des acquittements ?

f Combien d'octets émis sont acquittés en moyenne par ACK reçu ?

g Comment expliquez-vous la différence entre `204197 ACK (for 487292645 bytes)` et `17100 packets (2306201 bytes) received in sequence ?`

h Quelle est la taille moyenne des segments de données reçus ? Comparez avec la taille des segments émis. Que peut-on penser de la machine sur laquelle ces statistiques sont faites ?

i Combien de tentatives de connexion ont échoué ?

j Quel est le volume moyen de données émises et reçues par connexion ?

k Le paramètre RTT varie-t-il souvent ?

l Combien de fois le temporisateur a-t-il expiré parce que l'acquittement n'arrivait pas ?

m *Keepalive* est un temporisateur qui surveille le bon fonctionnement de la connexion en absence de trafic. Que pensez-vous de l'activité sur les connexions établies ?

Solution

a 352 383 segments (appelés ici paquets !) ont transporté 489 293 303 octets. Donc, la taille moyenne d'un segment est 489 293 303/352 383 = 1 388 octets (ou 1,355 Ko).

b La cause la plus vraisemblable des pertes est la non-fiabilité du ou des réseaux traversés : TCP attend les acquittements un certain temps et décide que les segments sont perdus quand l'acquittement n'est pas arrivé.

c Les paquets `window probe` et `window update` sont destinés à modifier la taille de la fenêtre pour un meilleur contrôle de flux et de congestion.

d Les paquets de contrôle sont ceux qui correspondent à l'ouverture et la fermeture des connexions ainsi que les acquittements de ces ouvertures et fermetures.

e Sur les 224 123 segments reçus, 204197 contiennent des acquittements. Donc la proportion est de : 204 197/224 123 = 91 %.

f Chaque acquittement reçu valide la réception par l'autre extrémité d'un certain nombre d'octets. Les 204 197 acquittements ont validé globalement 487 292 645 octets soit 487 292 645/204 197 = 2 386 octets (ou 2,33 Ko).

g Les 17 100 segments reçus en séquence correspondent aux données que la machine a reçues correctement. On constate ici que le trafic est dissymétrique puisque la machine reçoit peu de données et énormément d'acquittement.

h La taille moyenne des segments de données reçus est de 2 306 201/17 100 = 135 octets. La taille des segments émis est de 1,355 Ko soit 10 fois plus. On peut penser de la machine sur laquelle ces statistiques sont faites est un serveur, qui reçoit de la part de ses nombreux clients des requêtes simples et courtes et émet en réponse des messages longs. Ce pourrait être un serveur Web ou un serveur FTP, par exemple.

i La machine a reçu 356 demandes d'ouverture de connexion et en a accepté seulement 92. Il y a donc (356-92)/356 = 75 % tentatives de connexion qui ont échoué. Cela pourrait accréditer l'hypothèse qu'il s'agit d'un serveur FTP pour lequel il faut un « login » et un mot de passe pour pouvoir ouvrir une connexion.

j Le volume moyen de données émis par connexion est de 489 293 303/92 = 5 318 405 octets, soit 5 Mo. Le volume moyen de données reçu par connexion est de 2 306 201/92 = 24 Ko. Cela confirme bien le trafic dissymétrique de la machine.



k Il y a eu 176 379 mises à jour du paramètre RTT sur les 204 197 acquittements reçus. On voit que 86,3 % des acquittements correspondent à des acquittements normaux.

l Le temporisateur a expiré 21 fois parce que l'acquittement n'arrivait pas. Cela signifie que le RTT est bien adapté aux différentes connexions. D'autre part, ce n'est que parce que le segment a été effectivement perdu que son acquittement n'a pas été envoyé.

m Il n'y a eu aucune fermeture de connexion grâce au mécanisme *Keepalive* qui surveille le bon fonctionnement de la connexion en absence de trafic. On peut donc juger que l'activité sur les connexions établies est satisfaisante.

Exercice 8 Décodage de segment TCP

Énoncé

Décodez le segment TCP ci-après, donné en hexadécimal :

`00 15 0F 87 9C CB 7E 01 27 E3 EA 01 50 12 10 00 DF 3D 00 00`

Solution

`00 15`→ port source, ici 21 donc serveur FTP.

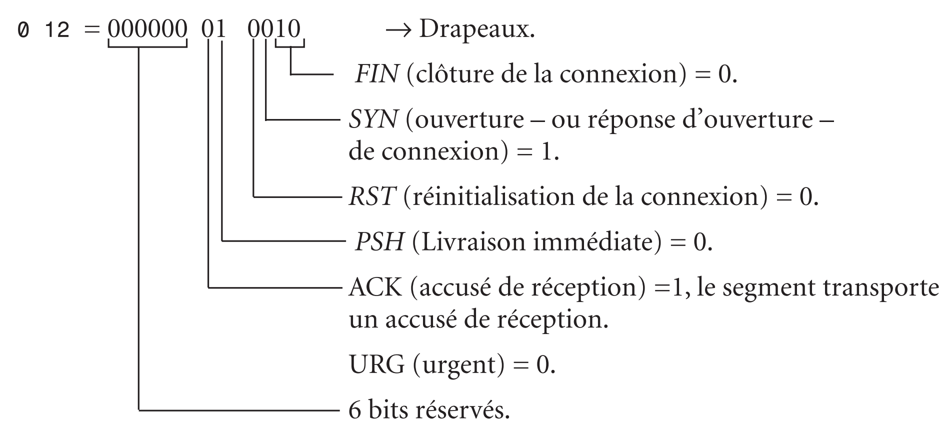
`0F 87`→ port destination 3975, port quelconque du client.

`9C CB 7E 01`→ Numéro de séquence (numéro du 1er octet émis).

`27 E3 EA 01`→ Numéro de séquence (numéro du 1er octet attendu en réception).

`5`→ Longueur de l'en-tête du segment (20 octets) : on peut en déduire que ce segment ne contient pas de données.

`0 12` = 000000 01 0010 → Drapeaux.



Les drapeaux SYN et ACK sont mis à 1.

`10 00`→ Taille de la fenêtre, ici *a priori* 4 096 octets.
C'est la quantité de données que l'émetteur est autorisé à envoyer sans accusé de réception.

`DF 3D`→ Bloc de contrôle d'erreur sur le segment entier.

`00 00`→ Pointeur vers les données urgentes (nul ici puisqu'il n'y a pas de données urgentes, bit URG = 0).

Fin du segment TCP (sans données).

Exercices

Bilan

- Ce segment contient deux drapeaux mis à 1, les drapeaux SYN et ACK. Il s'agit donc de la réponse positive à une demande d'ouverture de connexion. Les numéros de séquence initiaux viennent d'être choisis et le serveur, qui accepte la demande d'ouverture de connexion du client, fixe la taille de la fenêtre à 4 Ko. Il s'agit d'un serveur FTP, identifié par l'utilisation de son port bien connu 21.
- À l'analyse du segment seul, on ne peut pas savoir quelles sont les machines concernées par l'échange. On sait juste qu'il s'agit de l'initialisation d'un transfert de fichier FTP entre un client et un serveur.

Exercice 9 Décodage complet d'une trame

Énoncé

Décodez la trame Ethernet suivante (en hexadécimal) et en extraire toutes les informations possibles.

```
AA AA AA AA AA AA AA AB 08 00 20 0A 70 66 08 00 20 0A AC 96 08 00 45
00 00 28 A6 F5 00 00 1A 06 75 94 C0 5D 02 01 84 E3 3D 05 00 15 0F 87
9C CB 7E 01 27 E3 EA 01 50 12 10 00 DF 3D 00 00 20 20 20 20 20 20 9B
52 46 43
```

Solution

----------------------Début d'une trame Ethernet --

`AA AA AA AA AA AA AA AB` → Synchronisation (préambule et début de trame).

`08 00 20 0A 70 66` → adresse MAC destinataire (constructeur = 080020).

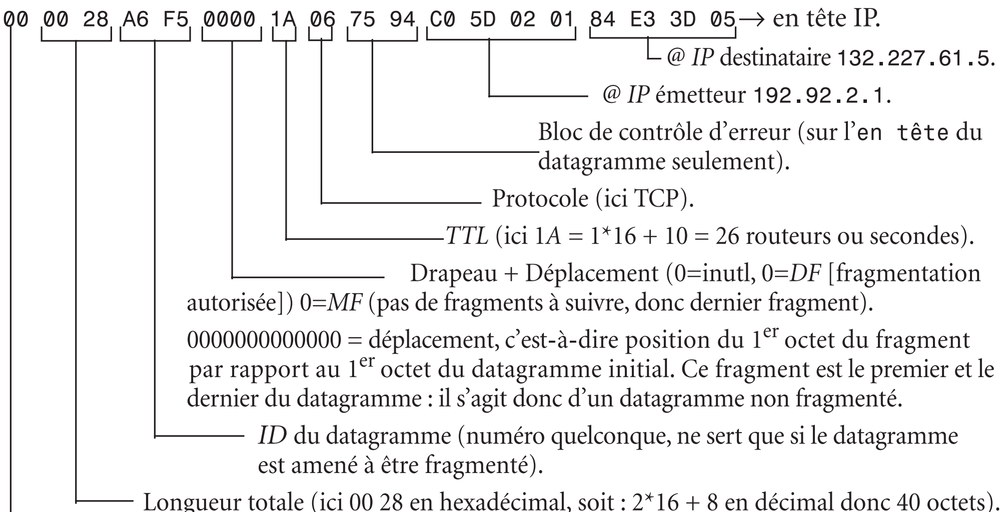
`08 00 20 0A AC 96` → adresse MAC émetteur (carte de même constructeur).

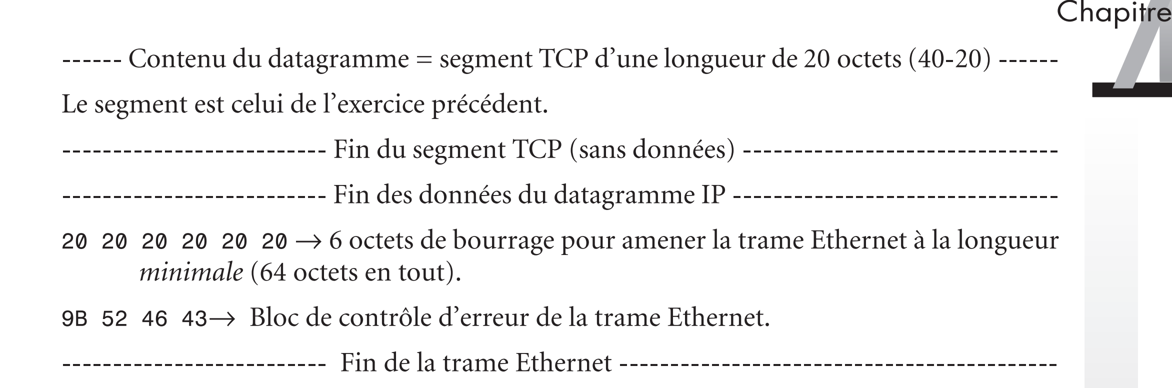
`08 00` → Type (ici IP)[si ce champ a une valeur inférieure à 1500, il s'agit d'une longueur].

------Début du contenu de la trame de longueur = 1 500 octets (ici datagramme IP) ------

`4`→ Version du protocole IP (IPv4).

`5`→ Longueur de l'en-tête (5*32 bits = 160 bits ou 5*4 octets = 20 octets).





------ Contenu du datagramme = segment TCP d’une longueur de 20 octets (40-20) ------

Le segment est celui de l’exercice précédent.

------------------------- Fin du segment TCP (sans données) ------------------------------

------------------------- Fin des données du datagramme IP -------------------------------

`20 20 20 20 20 20` → 6 octets de bourrage pour amener la trame Ethernet à la longueur *minimale* (64 octets en tout).

`9B 52 46 43`→ Bloc de contrôle d’erreur de la trame Ethernet.

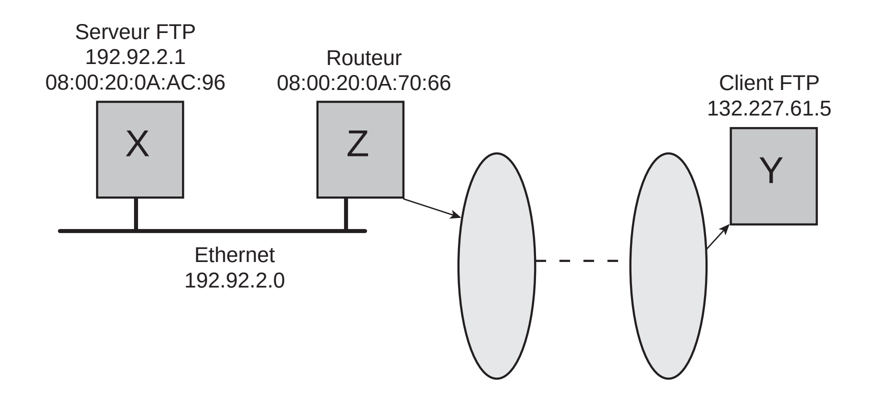
------------------------- Fin de la trame Ethernet --

Bilan

Cette trame Ethernet a été capturée dans un réseau que nous ne connaissons pas forcément. Trois possibilités sont à envisager :

a *Dans le réseau de l’émetteur de la trame (réseau de classe C 192.92.2.0).* Dans cette hypothèse, trois machines sont concernées par cet échange : la machine *X* d’adresse MAC 08 00 20 0A AC 96 et d’adresse IP 192.92.2.1. C’est un serveur FTP qui envoie une réponse positive à la demande d’ouverture de connexion que lui a faite la machine *Y*, client FTP d’adresse IP 132.227.61.5, se trouvant dans un autre réseau dont nous ne connaissons pas la technologie. La machine Z d’adresse MAC 08 00 20 0A 70 66 est donc le routeur de sortie du réseau 192.92.2.0. Elle est explicitement désignée comme destinataire de la trame Ethernet, puisque le datagramme IP que celle-ci contient doit sortir du réseau ! [Nous n’avons pas à connaître l’adresse IP du routeur]. La figure 7.7 illustre ce cas de figure.

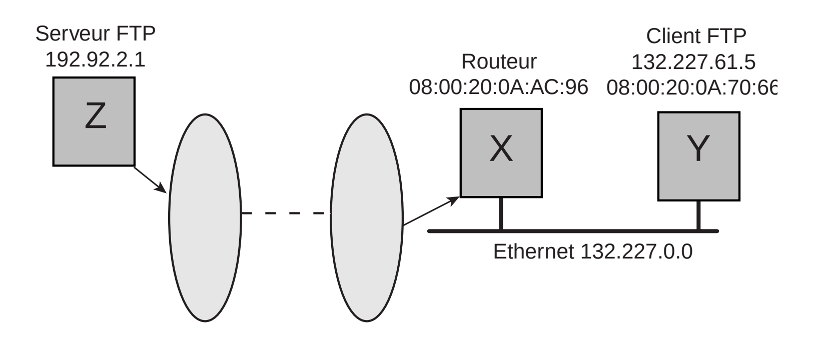
Figure 7.7 – ***Machines concernées par l’échange dans l’hypothèse a.***



b *Dans le réseau du destinataire (réseau de classe B 132.227.0.0).* De manière similaire, trois machines sont concernées par cet échange : la machine *X* d’adresse MAC 08 00 20 0A AC 96, qui est le routeur d’entrée du réseau en question. Ce routeur expédie une trame Ethernet à la machine *Y*, client FTP d’adresse MAC 08 00 20 0A 70 66 et d’adresse IP 132.227.61.5 se trouvant dans le réseau. La trame transporte l’accusé de réception à la demande de connexion qu’avait faite la machine *Y* (contenue dans une trame précédente que nous ignorons). La machine *Z* d’adresse IP 192.92.2.1 est le serveur FTP situé à l’extérieur du réseau, ce qui justifie que le datagramme soit relayé par le routeur d’entrée. La figure 7.8 illustre cette hypothèse.

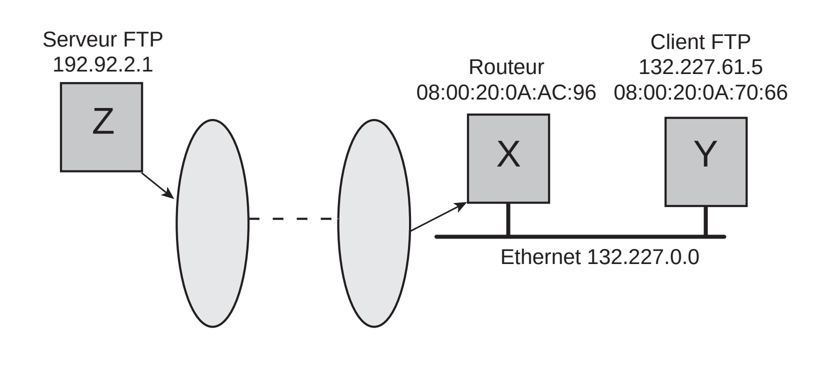
Bilan

Figure 7.8 – **Machines concernées par l'échange dans l'hypothèse b.**



c *Dans un réseau de transit, entre les deux réseaux concernés.* La capture a également pu être faite dans un réseau intermédiaire qui ne contient ni l'émetteur, ni le destinataire du datagramme. Les adresses MAC sont alors celles de deux routeurs intermédiaires, connus uniquement par leurs adresses MAC. Il y a donc quatre machines concernées par cet échange : la machine *X*, la machine *Y* et les routeurs d'entrée et de sortie du réseau de transit. La figure 7.9 montre les machines et les routeurs concernés dans l'hypothèse c.

Figure 7.9 – **Machines concernées par l'échange dans l'hypothèse c.**





Le routage

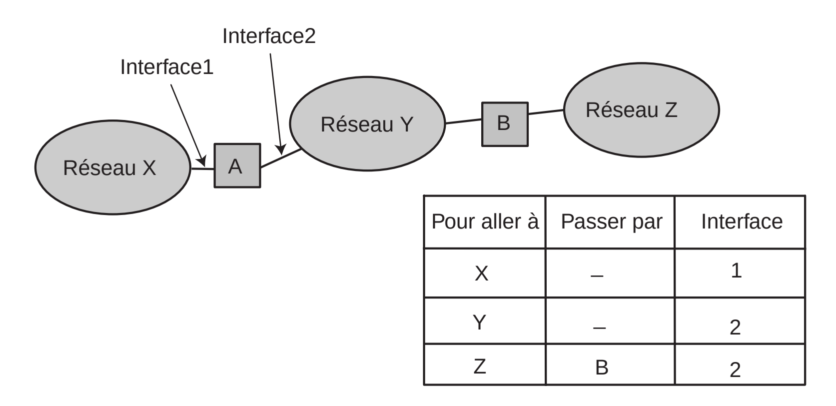
Nous avons vu au cours des chapitres précédents que l'acheminement des messages à travers un ou plusieurs réseaux nécessitait des connaissances sur le réseau et l'état de ses liaisons. Les équipements spécifiques, les routeurs, organisent cet acheminement. Ils utilisent pour cela des algorithmes classiques de recherche du meilleur chemin dans un graphe. Nous verrons que pour la mise en œuvre dans un grand réseau, il faut prévoir d'échanger entre ces routeurs des informations de contrôle dont le but est de construire pour chacun une table de routage. Cette table donne, pour chaque destination, la route à emprunter ainsi que son coût. Nous abordons les deux grandes familles d'algorithmes de routage et les protocoles associés pour véhiculer les informations entre les routeurs.

1 Objectifs du routage

Le but d'un protocole de routage est très simple : fournir l'information nécessaire pour effectuer un routage, c'est-à-dire la détermination d'un chemin à travers le réseau entre une machine émettrice et une machine réceptrice, toutes deux identifiées par leur adresse. Les protocoles de routage établissent des règles d'échange entre routeurs pour mettre à jour leurs tables selon des critères de coût comme, par exemple, la distance, l'état de la liaison, le débit. Ils améliorent ainsi l'efficacité du routage. La diversité des réseaux et de leurs services fait du routage un élément clé de leur bon fonctionnement. Il y a de très nombreux problèmes à résoudre. L'un des problèmes fondamentaux à éviter réside dans les boucles de routage (le message peut « tourner en rond » dans le réseau et ne jamais atteindre son destinataire). L'autre apparaît lorsqu'il y a une panne dans le réseau et qu'il faut optimiser le calcul des nouvelles routes : une fois la panne détectée, il faut transmettre l'information sur l'événement le plus rapidement possible pour que les différents routeurs recalculent par où faire passer leurs messages en contournant la liaison en panne.

Le premier protocole de routage sur Internet fut RIP (*Routing Information Protocol*). On lui préfère aujourd'hui une version plus élaborée, OSPF (*Open Shortest Path First*). Le premier s'appuie sur un algorithme de la famille *à vecteurs de distance*. Le second utilise un algorithmes de la famille *à état des liens*. Dans les deux cas, les algorithmes de base sont issus de la théorie des graphes (Bellman[1]-Ford, pour RIP, et Dijkstra[2], pour OSPF). La difficulté est de les mettre en œuvre dans des environnements réels avec efficacité, tout en minimisant la consommation des ressources réseau.

Les tables de routage s'accroissent au fur et à mesure de la taille du réseau. Cela augmente l'espace mémoire nécessaire dans les routeurs et les ressources processeur. D'autre part, cela contribue à diminuer les performances du réseau, puisque celui-ci doit propager un important trafic entre les routeurs eux-mêmes. On découpe alors le réseau en sous-ensembles régionaux. À l'intérieur d'une région, les tables de routage contiennent une entrée par routeur (voir figure 8.1). De cette façon, l'interconnexion de réseaux différents est aisée. Une hiérarchisation à plusieurs niveaux peut s'envisager pour les très gros réseaux, même si la distance parcourue entre régions n'est pas globalement optimale.



Pour aller à	Passer par	Interface
X	–	1
Y	–	2
Z	B	2

Figure 8.1

Un routeur A et sa table de routage.

Le réseau Internet est ainsi organisé comme une collection de « systèmes autonomes », et une seule autorité administre en général chacun d'entre eux. On appelle système autonome, ou SA, un ensemble de réseaux interconnectés partageant la même

1. Richard Bellman (1920-1984), mathématicien américain, connu pour la programmation dynamique.
2. Edsger Dijkstra (1930-2002), chercheur hollandais, l'un des plus influents dans le domaine de l'informatique, récompensé par le prix ACM Turing.

stratégie de routage : tous les routeurs internes à un système autonome respectent le même protocole de routage régi par une autorité administrative (un département responsable spécifique comme un fournisseur d'accès ou toute autre organisation) [voir figure 8.2].

On désigne comme protocole *interne* aux routeurs, le protocole de routage ou IGP (*Interior Gateway Protocol*) utilisé à l'intérieur d'un système autonome. Par opposition, un protocole *externe* appelé EGP (*Exterior Gateway Protocol*) transfère les informations de routage entre les différents systèmes autonomes.

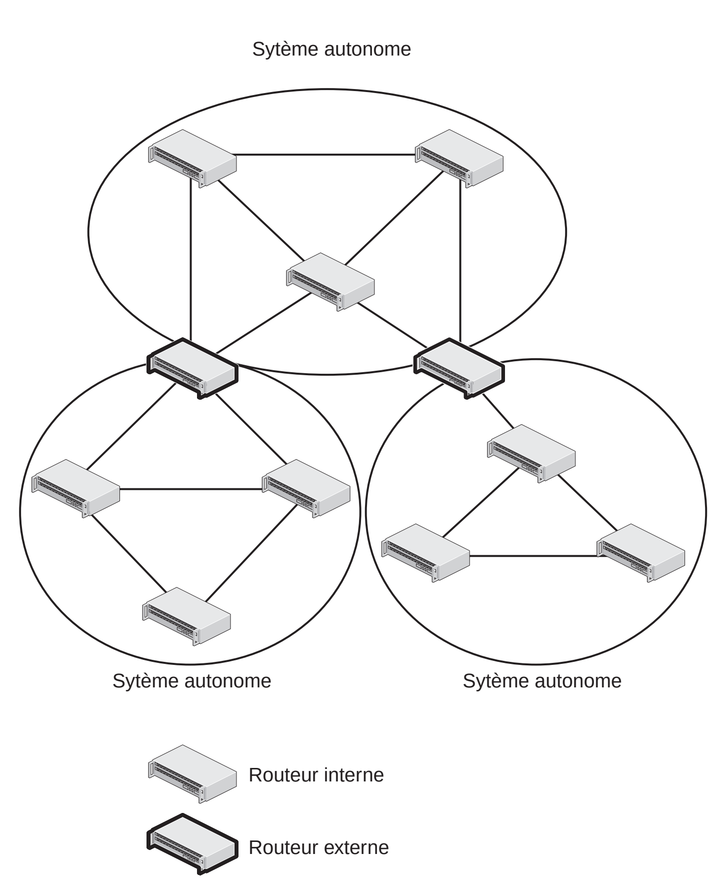


Figure 8.2

Systèmes autonomes et protocoles de routage internes et externes.

2 Principaux protocoles de routage

Nous décrivons succinctement dans cette section les protocoles de routage internes RIP et OSPF puis le protocole externe BGP.

2.1 RIP (*Routing Information Protocol*)

On a conçu RIP (*Routing Information Protocol*) pour fonctionner en tant que protocole interne dans des systèmes autonomes de taille modérée. Sa première version fut standardisée en 1988 (RFC 1058). La RFC 1723 propose des améliorations depuis 1994. RIP recherche le plus court chemin selon un critère de coût simple : le nombre de routeurs traversés. Cela revient à affecter un coût unitaire à la traversée de chaque routeur. RIP appartient à la famille des protocoles à vecteurs de distance, puisqu'il calcule la distance, en nombre de routeurs traversés, entre origine et destination.

Principe de fonctionnement

Le protocole est limité aux réseaux dont le plus long chemin (qu'on appelle couramment *le diamètre* du réseau) implique quinze routeurs au maximum. Il est mal adapté au traitement des boucles dans les chemins et utilise des mesures du coût des chemins (ou *métriques*) fixes pour comparer les routes alternatives. Les situations où les routes doivent être choisies en fonction de paramètres mesurés en temps réel comme un délai, une fiabilité ou une charge, se prêtent mal à ce type de traitement.

Un routeur RIP calcule des chemins pour différentes destinations, lesquelles sont spécifiées par leurs adresses IP, c'est-à-dire qu'une entrée dans la table peut représenter soit un réseau, soit un sous-réseau ou encore un nœud isolé. RIP ne spécifie pas le type de l'adresse : les routeurs découvrent la nature du destinataire en analysant les adresses transmises.

Les routeurs RIP sont *actifs* ou *passifs*. Actifs, ils transmettent et reçoivent les routes : ils diffusent leurs informations aux autres routeurs. Passifs, ils ne font qu'attendre la réception des informations. En fonction de celles-ci, ils calculent leurs tables de routage mais ne partagent pas les résultats de leurs calculs avec d'autres routeurs.

Le routeur RIP actif permet aux autres routeurs de mettre à jour leurs tables de routage toutes les 30 secondes. Si un routeur ne reçoit aucune mise à jour d'un autre routeur dans un délai de 180 secondes, il marque les routes desservies par ce dernier comme inutilisables. S'il n'y a aucune mise à jour après 240 secondes, le protocole RIP supprime toutes les entrées correspondant au routeur qui ne répond pas. Chaque diffusion RIP contient des paires adresses IP/nombre de routeurs à traverser (ou nombre de *sauts*). Comme le nombre de sauts est la seule mesure utilisée par le protocole, RIP ne garantit pas que le chemin sélectionné soit le plus rapide : un chemin court mais embouteillé peut être un mauvais choix par rapport à un chemin plus long mais totalement dégagé.

Lorsqu'un événement dans le réseau provoque un changement dans la table de routage d'un routeur actif, celui-ci envoie un message de mise à jour à ses voisins. Si cet événement a un impact sur les voisins, ceux-ci propagent l'information. On utilise une temporisation afin de stabiliser l'état du réseau et garantir que tous les messages de mise à jour ont été pris en compte avant de renvoyer une nouvelle mise à jour.

Variantes et améliorations

Des variantes procurent des améliorations au fonctionnement du protocole. Au lieu de diffuser le même message sur toutes les liaisons qui les relient, les routeurs composent des versions différentes de leurs informations, pour tenir compte des destinations atteintes *via* chaque liaison. Par exemple, si un routeur *A* envoie, *via B*, les messages à destination de *X*, c'est inutile pour *B* d'essayer d'atteindre *X via A*. Deux variantes sont possibles :

- Les routeurs ne donnent pas les informations sur les destinations atteintes à travers la liaison. Cette stratégie, dite « horizon partagé » (*Split-Horizon*), est la plus simple à implanter.
- Les routeurs indiquent dans leurs messages toutes les destinations possibles mais ils affectent une distance infinie pour celles situées en aval de ce nœud. Cette variante, dite « horizon partagé avec retour empoisonné » (*Split-Horizon with Poison-Reverse*), élimine immédiatement toute boucle de longueur 2.

Malgré ces améliorations, on ne supprime pas entièrement les risques de boucles.

Exemple

Soit un ensemble de trois routeurs *A*, *B*, *C*, illustrés à la figure 8.3, avec les tables de routage suivantes :

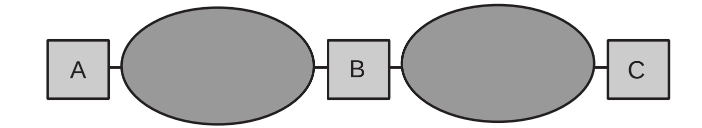
routage(A) = [(A, 0, –) ; (B, 1, B) ; (C, 2, B)].
routage(B) = [(A, 1, A) ; (B, 0, –) ; (C, 1, C)].
routage(C) = [(A, 2, B) ; (B, 1, B) ; (C, 0, –)].

Le triplet (*X*, *distance*, *Y*) signifie « pour aller à *X*, passer par *Y*, le chemin est de longueur *distance* ».

Si la liaison entre *B* et *C* tombe en panne, la table de *B* devient routage(*B*) = [(*A*, 1, *A*) ; (*B*, 0, –) ; (*C*, 16, *C*)]. En effet, 16 est considéré comme une destination inaccessible puisque le plus long chemin est de longueur 15. Quand *A* envoie sa table à *B*, celui-ci constate que *A* connaît une route de longueur 2 pour aller à *C*. Il met à jour sa propre table qui devient : (*B*) = [(*A*, 1, *A*) ; (*B*, 0, –) ; (*C*, 3, *A*)]. Cela crée une boucle puisque, pour aller à *C*, le routeur *A* envoie les messages vers *B* et le routeur *B* les renvoie vers *A*…

Figure 8.3

Trois routeurs et une boucle potentielle.



La nouvelle version RIPv2 fonctionne sur le même principe. Ce protocole véhicule simplement des informations supplémentaires, comme une étiquette qui fait la distinction entre les routes internes apprises nativement par RIP et les routes externes apprises par un autre protocole de routage. Il transporte de même le masque de sous-réseau qui permet d'affiner les routes avec la connaissance des sous-réseaux ou de l'*agrégation des adresses*[3]. Ces informations allègent la taille des tables de routage et participent à une meilleure efficacité du routage.

Remarque

Du point de vue de l'architecture en couches, le routage RIP est une application ; il utilise UDP comme protocole de transport avec le numéro de port 520. Un routeur RIP est donc une machine avec une architecture complète de protocoles, y compris Transport et Application.

D'autres fonctionnalités ont été proposées pour améliorer le fonctionnement du routage. On a cherché, par exemple, des moyens de rompre la *synchronisation* (les routeurs se mettent tous à diffuser leurs informations au même moment, et provoquent une rafale de trafic qui peut être insupportable). Puisque RIP utilise le protocole de transport UDP, on a imaginé des moyens d'acquitter de manière sûre la mise à jour, mais aussi de supporter plusieurs métriques, de rompre les boucles… Finalement, on a retenu une autre solution dans Internet : celle d'utiliser les protocoles à état des liens comme OSPF, considérés comme supérieurs aux protocoles à vecteurs de distance et que nous présentons à la section suivante.

2.2 OSPF (*Open Shortest Path First*)

L'algorithme SPF (*Shortest Path First*) calcule le plus court chemin vers toutes les destinations de la zone ou du système autonome, en partant du routeur où s'effectue le calcul (à partir de sa base de données topologiques). Il utilise un algorithme conçu par Dijkstra et calcule le plus court chemin, selon un critère de coût où interviennent de multiples paramètres : l'état des liens, l'encombrement du réseau et des mémoires des routeurs intermédiaires.

3. Cette expression désigne le regroupement du plus grand nombre de bits communs à deux ou plusieurs adresses IP. Par exemple, les adresses réseau : `195.67.65.0 et 195.67.127.0` ont leurs 17 premiers bits en commun. Pour un routeur donné, si le chemin pour atteindre les deux réseaux passe par le même routeur suivant, sa table de routage ne contiendra qu'une seule entrée pour les deux réseaux.

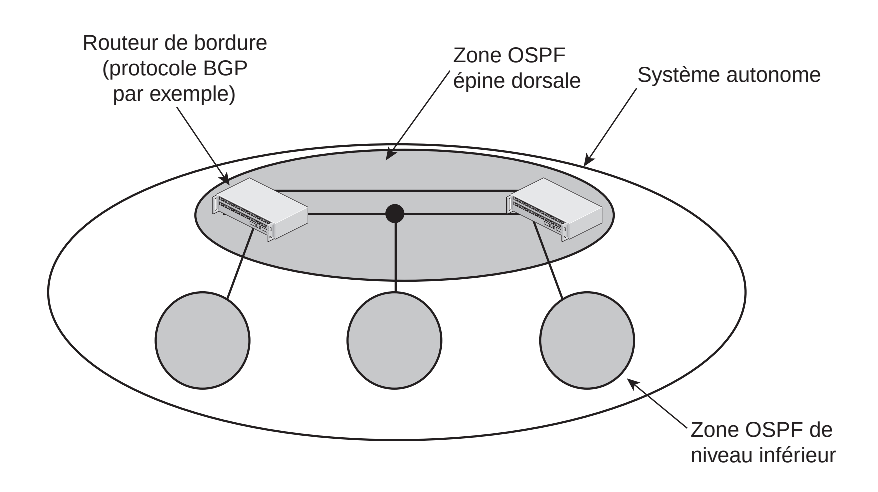
Principe de fonctionnement

Le calcul du plus court chemin est effectué de manière indépendante par tous les routeurs internes d'un système autonome SA. Grâce à cet algorithme, un routeur peut connaître le prochain routeur qui transmettra le message : il trouve les plus courts chemins (en termes de coût) d'un point à un autre, pour que le message arrive de manière optimale à son destinataire, puis il effectue la mise à jour de sa table de routage. Chaque mise à jour de la base de données entraîne celle de la table de routage. Il y a, comme précédemment, communication entre les routeurs. Celle-ci est régie par le protocole OSPF.

Ce protocole définit les règles et les formats de messages entre routeurs OSPF internes à un système autonome. Il a la particularité de s'appuyer directement sur IP (et non sur UDP comme le protocole RIP). C'est une nette amélioration, car le routage devient un traitement interne à la couche réseau et n'est plus considéré comme une application. De plus, le fonctionnement d'OSPF est optimisé si le SA est découpé en zones ; OSPF prévoit un découpage avec une hiérarchie à deux niveaux de zones. La zone de niveau le plus élevé est l'*épine dorsale* (ou *backbone zone*). Elle interconnecte les « routeurs de bordure » (*edge routers*). À l'intérieur de chaque zone du second niveau, les routeurs ne connaissent – et ne diffusent – que des informations internes à leur zone. Un routeur de bordure dans chaque zone assure le lien avec l'épine dorsale, comme le montre la figure 8.4.

Figure 8.4

Hiérarchie d'organisation des zones OSPF.



Messages du protocole OSPF

On distingue cinq messages OSPF : *hello*, *description de base de données*, *requête d'état de lien*, *mise à jour d'état de lien*, *acquittement d'état de lien*. Ils transportent des informations sur l'état des liaisons du SA et servent à déterminer une fonction de coût plus efficace que dans RIP.

Un routeur OSPF émet des messages hello à intervalles réguliers (environ toutes les dix secondes), sur chacune de ses interfaces. Ces messages établissent les relations d'adjacence[4] avec les routeurs directement liés à l'émetteur de ces messages. Les routeurs qui les ont reçus vérifient que les chemins restent disponibles.

Sur un réseau possédant au moins deux routeurs, on élit un *routeur désigné*, c'est-à-dire le responsable qui échange avec les routeurs des réseaux voisins. Il s'occupe de la distribution des messages de mise à jour d'état de lien. Son choix se fait sur la base de la plus petite

4. Dans la théorie des graphes, deux nœuds sont adjacents s'ils sont directement reliés. Ici, la notion d'adjacence est légèrement différente, puisqu'elle ajoute une règle supplémentaire : le routeur désigné est adjacent à tous les autres. C'est l'efficacité qui prévaut : dans un réseau local, il est inutile que tous les routeurs participent au routage. Seul l'un entre eux est le routeur désigné, tous les autres lui sont adjacents.



adresse IP parmi les routeurs susceptibles d'assumer ce rôle. Deux routeurs *R1* et *R2* établissent une relation d'adjacence si et seulement s'ils sont reliés par un lien direct ou si l'un d'entre eux est routeur désigné. Lorsqu'une nouvelle adjacence s'établit entre deux routeurs, ils synchronisent leurs bases de données d'état des liens par le message description de base de données.

Il est très important de protéger la base de données contre des erreurs (accidentelles ou dues à la malveillance), pour garantir la cohérence des calculs de chemins. Pour cela, on a prévu plusieurs précautions : acquittement, transmission sécurisée et temporisation des messages sur chaque liaison. Le message *acquittement d'état de lien* accuse réception d'une mise à jour : le routeur qui a envoyé ses indications de coût vers ses voisins sait que le message est bien parvenu. La transmission des messages de *description de base de données* est sécurisée : chaque enregistrement est protégé par un total de contrôle et tous les messages sont authentifiés. Cela évite d'éventuels messages qui contiendraient des informations erronées (éventuellement malveillantes, afin de détourner le trafic dans le réseau, par exemple). Enfin, on associe une temporisation à tout enregistrement : l'information contenue dans l'entrée de la table de routage sera supprimée si elle n'a pas été rafraîchie récemment : on préfère n'avoir aucune information momentanément plutôt qu'une information ancienne et inutilisable. Quand un routeur constate qu'une ou plusieurs des entrées de sa base de données sont périmées, il envoie une *requête d'état de lien* aux routeurs voisins pour faire la mise à jour des données.

OSPF est aujourd'hui le protocole interne le plus utilisé dans Internet. La qualité des informations transportées et la sécurité associée sont ses principaux atouts. Le fait que le routage reste interne à la couche réseau est un élément d'efficacité supplémentaire.

2.3 BGP (*Border Gateway Protocol*)

BGP est le protocole de routage interdomaine utilisé actuellement partout dans le monde. Conçu pour échanger des informations de routage entre systèmes autonomes, il est défini dans les RFC 1771 et 1774. Pour BGP, les différents réseaux sont organisés en SA, reliés par une ou plusieurs liaisons. Au sein d'un SA, le routage est calculé avec l'un des protocoles précédents (RIP ou OSPF). BGP intervient lorsque la route doit emprunter plusieurs SA.

Dans un système autonome, il y a un ou plusieurs routeurs de bordure qui dialoguent avec le ou les routeurs de bordure des SA voisins. Lorsqu'il n'y a qu'un seul routeur de bordure, le SA est vu comme un cul-de-sac : il ne peut pas être un SA de transit pour des messages sur l'interconnexion. Il est alors ignoré de BGP. Quand il y en a plusieurs, le SA est un SA de transit : des messages sur l'interconnexion peuvent y entrer par l'un des routeurs de bordure et ressortir par l'autre. Il faut toutefois ajouter que certains SA interdisent le transit interne (pour des raisons politiques ou commerciales, par exemple). Ils sont encore ignorés de BGP.

BGP ne prend donc en compte que les SA où le transit est autorisé (réseaux fédérateurs d'Internet, par exemple, ou réseaux d'opérateurs, moyennant des accords financiers). Les routeurs de bordure de ces SA sont appelés *routeurs BGP*. Ils calculent des routes avec un algorithme à vecteurs de distance. À la différence de RIP, ils mémorisent la totalité du chemin et non seulement le premier routeur du chemin. Ils échangent donc des informations complètes, ce qui est possible car le graphe BGP est de petite taille.

Les routeurs BGP communiquent par échange de messages transportés par des connexions TCP permanentes sur le port 179. Un routeur BGP est une machine dotée d'une architecture de communication complète, car le routage entre SA est considéré comme une application exigeant une grande fiabilité de communication.

2.4 VECTEURS DE DISTANCE OU ÉTAT DES LIENS

Nous avons vu qu'il y avait deux familles d'algorithmes (à vecteurs de distance et à état des liens). La première calcule le meilleur chemin selon sa longueur (généralement exprimée en nombre de routeurs traversés). La seconde calcule le meilleur chemin selon une fonction de coût (le meilleur délai de traversée par exemple). Étudions quelques éléments d'analyse et de comparaison de ces deux familles : rapidité de convergence de l'algorithme, possibilités de métriques différentes, choix d'un chemin parmi plusieurs équivalents, utilisation de routes externes. On considère généralement qu'un protocole à état des liens offre plusieurs avantages par rapport à un protocole à vecteurs de distance.

- *Convergence rapide et sans boucle de l'algorithme.* Dans un algorithme à vecteurs de distance, le nombre d'itérations est proportionnel au nombre de routeurs. Dans le pire cas, il est égal au nombre de routeurs moins 1. Dans un algorithme à état des liens, la convergence s'établit en deux phases : transmission rapide des nouvelles informations puis calcul local du chemin. De plus, cette méthode évite les boucles, puisque tous les chemins calculés sont sains.
- *Métriques multiples.* Alors qu'il est difficile d'utiliser des métriques trop fines dans les algorithmes à vecteurs de distance, on peut supporter plusieurs métriques en parallèle, sans ralentir la convergence, dans les protocoles à état des liens. Cela provient du fait que la topologie est complètement connue pendant le calcul des chemins. On peut donc choisir la meilleure route en fonction de critères différents, en appliquant des métriques différentes. Les algorithmes à état des liens sont les premiers à offrir un routage en fonction de la qualité de service requise par l'utilisateur.
- *Chemins multiples.* Dans un protocole à vecteurs de distance, le choix d'un chemin parmi plusieurs se fait au hasard de la chronologie des échanges de messages. De plus, il n'est prévu qu'un seul routeur suivant dans la table de routage. Moyennant une légère modification de l'algorithme, les protocoles à état des liens peuvent tolérer des chemins multiples. On peut ainsi répartir le trafic entre plusieurs chemins équivalents en termes de coûts. L'équilibrage du trafic dans le réseau est une valeur ajoutée considérable, car elle contribue à la fluidité de la circulation des données et permet un réel contrôle de congestion.
- *Routes externes.* Les problèmes de routage que nous avons évoqués ne concernent que l'acheminement des données dans un réseau (considéré comme un ensemble homogène de stations et de routeurs). Une route externe est une route qui passe par d'autres zones ou d'autres réseaux que celui dans lequel on se trouve. Dans les grands réseaux – et plus encore dans Internet –, la connectivité se réalise à travers plusieurs points d'accès à différents réseaux de transit. Les éléments du choix des routes deviendraient trop complexes dans un protocole à vecteurs de distance : il faudrait prendre en compte plusieurs points d'accès, plusieurs prestataires de services, utiliser une route par défaut… Avec la possibilité d'utiliser des métriques multiples, les calculs de chemins intégrant des routes externes se font plus naturellement dans les protocoles à état des liens.

3 Routage et évolution des réseaux

Les services offerts dans les réseaux évoluent. La diffusion partielle ou totale des messages et certaines architectures comme les réseaux *peer-to-peer* posent des problèmes spécifiques. En outre, la présence d'utilisateurs mobiles est un nouveau défi pour le routage.

3.1 Diffusion

La diffusion générale (*broadcast*) suppose que le message est destiné à toutes les stations du réseau. La diffusion restreinte (*multicast*) suppose que le message est à transmettre à une liste de destinataires ou à un groupe d'utilisateurs, identifiés par une seule adresse dite *adresse de groupe*. On peut envisager plusieurs méthodes pour diffuser un même message à plusieurs destinataires : envoyer autant de messages que de destinataires, inonder le réseau avec des copies du message, calculer et gérer un arbre couvrant pour atteindre tous les destinataires.

- *Envoyer autant de messages que de destinataires.* Cette solution simpliste entraîne un gaspillage des ressources dans le réseau et nécessite le maintien d'une liste complète de tous les destinataires.
- *Inondation (envoi d'une copie du message sur chaque route).* Le nombre de messages et les ressources consommées sont excessifs… Il n'y a pas besoin de contrôler la liste des destinataires mais l'arrêt de l'inondation n'est pas un problème simple !
- *Routage multidestination.* Dès qu'un message arrive dans un routeur, ce dernier examine toutes les destinations pour déterminer les interfaces de sortie requises. Il génère autant de copies que nécessaire, en explicitant les destinataires concernés sur chaque interface. Après la traversée d'un certain nombre de routeurs, on se retrouve avec des messages qui n'ont plus qu'une seule destination.
- *Calcul d'un arbre couvrant par le premier routeur.* Cette méthode consiste à utiliser une variante de l'algorithme de *Spanning Tree* que nous avons vu lors du fonctionnement des ponts, au chapitre 5. Si le routeur connaît sur l'ensemble de ses liaisons celles qui font partie de l'arbre couvrant, il copie le message sur toutes les liaisons concernées (sauf celle d'où provient le message). Ce mécanisme est efficace mais il nécessite la connaissance de l'arbre couvrant.
- *Utilisation de l'algorithme RFP (Reverse Forwarding Path).* Lorsqu'un message arrive sur un routeur, ce dernier détermine si le message a suivi le chemin que lui-même utilise pour joindre l'émetteur. Si c'est le cas, il est vraisemblable que ce message a emprunté le plus court chemin depuis l'émetteur et qu'il s'agit du paquet d'origine. Le routeur envoie alors une copie du message sur toutes ses interfaces (sauf sur celle d'où provient le message). Dans le cas contraire, il considère le message comme un doublon et le rejette.

Pour les communications de groupe, il faut adapter les méthodes précédentes et faire vivre le groupe : un utilisateur nouveau peut se joindre à un groupe ou il peut en partir. Cela suppose l'utilisation d'un système de gestion de groupes qui crée ou supprime des groupes, autorise un utilisateur à rejoindre ou quitter un groupe, etc. Ces tâches de gestion des groupes sont transparentes pour l'algorithme de routage. Elles sont prises en compte par un protocole comme IGMP (*Internet Group Management Protocol*, RFC 2236). En revanche, les routeurs doivent savoir à quels groupes appartiennent les utilisateurs. Ils sont soit directement informés par les hôtes, soit les processus IGMP sont interrogés périodiquement. Les informations de routage sont alors communiquées aux voisins du routeur et se propagent ainsi à travers le réseau.

3.2 Utilisateurs mobiles

Les utilisateurs d'ordinateurs portables souhaitent lire leur courrier ou accéder au système de fichiers traditionnel, même quand ils sont en déplacement. Avant de router un message vers eux, il faut d'abord les localiser. Par définition, les équipements *fixes* ne se déplacent jamais. Ils sont connectés au réseau par des liaisons filaires (câbles en cuivre ou fibres optiques) ; les équipements *migrateurs* changent de site de temps à autre et n'utilisent le réseau que lorsqu'ils y sont physiquement raccordés. Enfin, les équipements *itinérants* se déplacent et

accèdent en permanence ou de façon intermittente au réseau. On peut même envisager des réseaux où tous les équipements sont mobiles ! On considère généralement les équipements migrateurs et les équipements itinérants comme des équipements *mobiles*.

Tous les équipements mobiles disposent d'un *site de domiciliation permanente* qui ne change jamais. Ils possèdent une adresse permanente qui ne suffit plus pour les localiser. L'envoi de messages pour des utilisateurs mobiles suppose un routage qui fonctionne d'après leurs adresses permanentes, quel que soit le lieu où ils se trouvent. On découpe généralement l'espace de déplacement en petites unités appelées *zones* (une *cellule radio*, par exemple). Chaque zone possède un ou plusieurs *agents extérieurs* (*foreign agents*) qui assurent le suivi des équipements mobiles se trouvant momentanément dans la zone. Une zone dispose en outre d'un *agent de domiciliation* (*home agent*) qui gère les équipements domiciliés dans la zone mais actuellement présents dans une autre zone.

Pour chaque mobile, un dialogue s'instaure entre son agent de domiciliation et les agents extérieurs des différentes zones qu'il visite. Ce dialogue dépend du parcours du mobile et de sa vitesse de déplacement. Les routeurs obtiennent, grâce à ce dialogue, les informations qui localisent le mobile dans la zone où il se trouve. Ils peuvent ainsi calculer la route pour l'atteindre.

3.3 Réseaux *peer-to-peer*

Un grand nombre de personnes possédant des connexions Internet permanentes souhaitent communiquer pour partager directement leurs ressources au moyen de réseaux peer-to-peer. Cette technologie est utilisée dans de nombreuses applications intéressantes et légales. Elle est souvent associée à l'idée de copies illégales de fichiers audio ou vidéo.

Il existe deux types principaux de réseaux *peer-to-peer* : les architectures totalement distribuées et les architectures hybrides. Dans la première, tous les équipements sont symétriques (on ne parle plus de client ni de serveur, puisque tout équipement est à la fois client et serveur). Il n'y a pas de contrôle central, ni de rapports hiérarchiques entre équipements. Il faut donc localiser un équipement en absence d'annuaire centralisé, car personne ne souhaite héberger et maintenir la base de données des ressources et de tous les équipements concernés. Cela suppose l'utilisation d'algorithmes particuliers, puisqu'il faut d'abord connaître l'équipement qui offre la ressource recherchée puis trouver le chemin pour l'atteindre. Dans la seconde, une station gère la base de données qui permet la localisation des ressources et des équipements et facilite donc le routage au sein du réseau.

Résumé

L'acheminement des messages à travers un ou plusieurs réseau(x) nécessite des connaissances sur le réseau et l'état de ses liaisons. Les routeurs organisent cet acheminement. Ils utilisent pour cela des algorithmes classiques de recherche du meilleur chemin dans un graphe. Il y a deux grandes familles d'algorithmes de routage : ceux à vecteurs de distance calculent le plus court chemin au sens du nombre de routeurs traversés ; ceux à état des liens estiment le coût des différents tronçons du réseau. Leur mise en œuvre dans un grand réseau n'est pas simple. Les routeurs échangent entre eux des informations de contrôle dont le but est la construction d'une table de routage pour chacun. Cette table donne, pour chaque destination, la route à emprunter ainsi que son coût. Pour faciliter les opérations de routage, les réseaux sont découpés en systèmes autonomes et le problème est d'abord résolu à l'intérieur d'un système puis entre deux systèmes, éventuellement avec des protocoles différents pour transporter les informations de routage.



Problèmes et exercices

Exercice 1 Table de routage

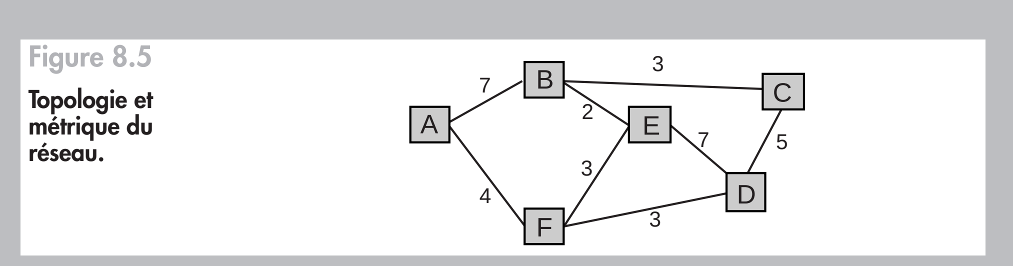
Énoncé Établissez la table de routage du nœud *E* de ce réseau, en minimisant le coût des liaisons. Vous supposerez que la topologie entière du réseau est connue.

a Vous utilisez un algorithme à vecteurs de distance.

b Vous utilisez un algorithme à état des liens qui s'appuie sur la métrique indiquée à la figure 8.5.

Figure 8.5

Topologie et métrique du réseau.



Solution

a La table de routage du nœud *E* peut être, par exemple :

routage(*E*) = [(*A*, *B*) ; (*B*, *B*) ; (*C*, *B*) ; (*D*, *D*) ; (*E*, –) ; (*F*, *F*)] où le couple (*A*, *B*) signifie : pour aller à *A*, il faut passer par *B*.

Il y a deux chemins de longueur 2 pour aller de *E* à *A*, celui qui passe par *B* et celui qui passe par *F*. Nous avons retenu celui qui correspondant à la plus petite lettre dans l'ordre alphabétique. De même pour le chemin de *E* à *C*.

b Avec un algorithme à état des liens, il faut comparer les différents chemins. Le chemin *E-B-A* est de coût 7 + 2 = 9 alors que *E-F-A* est de coût 3 + 4 = 7. Ce dernier est meilleur. Remarquons qu'un chemin long comme *E-F-D* est meilleur que le chemin direct *E-D* puisque 3 + 3 = 6 est meilleur que 7. L'algorithme de Dijkstra doit donc explorer tous les chemins.

On procède par étapes. Cherchons les chemins de longueur 1. On trouve E-B = 2, E-D = 7, E-F = 3. Cherchons maintenant les chemins plus longs à partir du lien le plus prometteur, c'est-à-dire E-B. On trouve E-B-A = 2 + 7 = 9 et E-B-C = 2 + 5 = 7. Cherchons ensuite les chemins plus longs à partir du lien prometteur suivant c'est-à-dire E-F. On trouve E-F-A = 3 + 4 = 7, meilleur que l'information précédemment calculée : cette dernière est effacée, on ne conserve que le meilleur chemin. De même E-F-D = 3 + 3 = 6 est meilleur que E-D précédemment calculé à 7. On continue ainsi en explorant les chemins à partir du lien prometteur suivant, ici E-C, etc.

La table de routage de *E* est finalement :

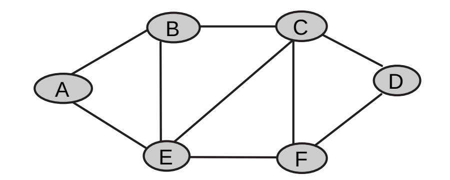
routage(E) = [(A, F) ; (B, B) ; (C, B) ; (D, F) ; (E, –) ; (F, F)].

EXERCICE 2 ROUTAGE AVEC RIP

Énoncé On considère le système autonome illustré à la figure 8.6, constitué de 6 routeurs nommés *A, B, C D, E* et *F*. On utilise le protocole RIP comme protocole de routage interne pour ce système autonome avec une périodicité d'envoi de messages de 30 secondes. Le coût de chaque saut est toujours 1. Une table de routage sera notée comme un ensemble d'entrées (*X, distance, Y*) dans lequel *X* est la destination qu'on cherche à atteindre, *distance* est la distance actuellement connue en passant par *Y* qui est donc le routeur suivant sur le chemin. Pour le routeur *X*, l'entrée correspondant à la destination *X* sera notée (*X, 0,* –) soit une distance nulle sans routeur intermédiaire.

Figure 8.6

Système autonome à 6 routeurs.



a Établissez les tables de routage de chacun des 6 routeurs. L'ordre dans lequel circulent les messages du protocole RIP a-t-il de l'importance ? Combien de temps faut-il pour que l'algorithme converge ?

b La liaison entre *B* et *C* tombe en panne. Le routeur *B* détecte cette rupture et met à jour sa table en supposant que *C* se trouve maintenant à une distance 16 ; de même pour le routeur *C*. Montrez comment l'information se propage aux autres routeurs. Combien de temps faut-il pour que tous les routeurs aient recalculé leur table de routage ?

Solution **a** À l'état initial, chaque routeur ne connaît que ses voisins directs. La table de routage de A est donc :

routage(A) = [(A, 0, –) ; (B, 1, B) ; (E, 1, E)].

De même les tables de routage des autres routeurs sont :

routage(B) = [(A, 1, A) ; (B, 0, –) ; (C, 1, C) ; (E, 1, E)].

routage(C) = [(B, 1, B) ; (C, 0, –) ; (D, 1, D) ; (E, 1, E) ; (F, 1, F)].

routage(D) = [(C, 1, C) ; (D, 0, –) ; (F, 1, F)].

routage(E) = [(A, 1, A) ; (B, 1, B) ; (C, 1, C) ; (E, 0, –) ; (F, 1, F)].

routage(F) = [(C, 1, C) ; (D, 1, D) ; (E, 1, E) ; (F, 0, –)].

À la date T = 30 secondes, chaque routeur envoie sa table à ses voisins. Nous traiterons les événements dans l'ordre alphabétique.

Quand A envoie routage(*A*) = [(*A*, 0, –) ; (*B*, 1, *B*) ; (*E*, 1, *E*)] à *B*, celui-ci constate qu'il n'y a aucune nouvelle entrée et que pour celles qu'il a déjà, les informations envoyées par *A* ne sont pas plus intéressantes que les siennes. De même pour *E* quand il reçoit routage(*A*) = [(*A*, 0, –) ; (*B*, 1, *B*) ; (*E*, 1, *E*)].

Maintenant traitons l'envoi par *B* de routage(*B*) = [(*A*, 1, *A*) ; (*B*, 0, -) ; (*C*, 1, *C*) ; (*E*, 1, *E*)] à ses voisins *A*, *C* et *E*.



A apprend que *B* connaît une route de distance 1 pour atteindre *C*, il ajoute donc dans sa table une nouvelle entrée (*C*, 2, *B*) : *C* est à une distance 1 de *B* ou *B* est à une distance 1 de *A* donc *C* est globalement à une distance 1 + 1 = 2 en passant par *B*. Les autres informations envoyées par *B* ne changent rien, la nouvelle table de *A* est maintenant :

routage(A) = [(A, 0, –) ; (B, 1, B) ; (C, 2, B) ; *(E, 1, E)].*

C apprend que *B* connaît une route de distance 1 pour atteindre *A*, il ajoute donc dans sa table une nouvelle entrée (*A*, 2, *B*) : *A* est à une distance 1 de *B* ou *B* est à une distance 1 de *C* donc A est globalement à une distance 1 + 1 = 2 en passant par *B*. Les autres informations envoyées par *B* ne changent rien, la nouvelle table de *C* est maintenant :

routage(C) = [(A, 2, B) ; *(B, 1, B) ; (C, 0, –) ; (D, 1, D) ; (E, 1, E) ; (F, 1, F)].*

Pour E, l'envoi de B n'apporte rien, sa table reste inchangée.

Maintenant traitons l'envoi par *C* de routage(*C*) = [(*B*, 1, *B*) ; (*C*, 0, –) ; (*D*, 1, *D*) ; (*E*, 1, *E*) ; (*F*, 1, *F*)] à ses voisins *B*, *D*, *E* et *F*. (Remarque : il s'agit bien de la table initiale de *C* et non de celle qui a été mise à jour ci-dessus ; par contre les mises à jour se font sur les tables déjà modifiées.)

B apprend que *C* connaît une route de distance 1 pour atteindre *D* et *F*, il ajoute donc dans sa table deux nouvelles entrées (*D*, 2, *C*) et (*F*, 2, *C*). Les autres informations envoyées par *C* ne changent rien, la nouvelle table de *B* est maintenant :

routage(B) = [(A, 1, A) ; (B, 0, –) ; (C, 1, C) ; (D, 2, C) ; *(E, 1, E) ;* (F, 2, C)*].*

La nouvelle table de *D* devient :

routage(D) = [(B, 2, C) ; *(C, 1, C) ; (D, 0, –) ;* (E, 2, C) ; *(F, 1, F)].*

et de même pour *E* et *F* :

routage(E) = [(A, 1, A) ; (B, 1, B) ; (C, 1, C) ; (D, 2, C) ; *(E, 0, –) ; (F, 1, F)].*

routage(F) = [(B, 2, C) ; *(C, 1, C) ; (D, 1, D) ; (E, 1, E) ; (F, 0, –)].*

L'envoi par *D* de sa table routage(*D*) = [(*C*, 1, *C*) ; (*D*, 0, –) ; (*F*, 1, *F*)] n'apporte rien à *C* ni à *F*.

L'envoi par *E* de sa table routage(*E*) = [(*A*, 1, *A*) ; (*B*, 1, *B*) ; (*C*, 1, *C*) ; (*E*, 0, –) ; (*F*, 1, *F*)] provoque l'apparition de l'entrée *F* dans la table de *A* et de l'entrée *A* dans la table de *F*.

routage(A) = [(A, 0, –) ; (B, 1, B) ; (C, 2, B) ; (E, 1, E) ; (F, 2, E)*]*

routage(F) = [(F, 2, E) ; *(B, 2, C) ; (C, 1, C) ; (D, 1, D) ; (E, 1, E) ; (F, 0, –)].*

Enfin, l'envoi par *F* de sa table routage(*F*) = [(*C*, 1, *C*) ; (*D*, 1, *D*) ; (*E*, 1, *E*) ; (*F*, 0, –)] ne change rien.

À la fin de cette étape, les tables de routage sont donc les suivantes :

routage(A) = [(A, 0, –) ; (B, 1, B) ; (C, 2, B) ; *(E, 1, E)].*

routage(B) = [(A, 1, A) ; (B, 0, –) ; (C, 1, C) ; (D, 2, C) ; *(E, 1, E) ;* (F, 2, C)*].*

routage(C) = [(A, 2, B) ; *(B, 1, B) ; (C, 0, –) ; (D, 1, D) ; (E, 1, E) ; (F, 1, F)].*

routage(D) = [(B, 2, C) ; *(C, 1, C) ; (D, 0, –) ;* (E, 2, C) ; *(F, 1, F)].*

routage(E) = [(A, 1, A) ; (B, 1, B) ; (C, 1, C) ; (D, 2, C) ; *(E, 0, –) ; (F, 1, F)].*

routage(F) = [(F, 2, E) ; (B, 2, C) ; *(C, 1, C) ; (D, 1, D) ; (E, 1, E) ; (F, 0, –)].*

On constate que les chemins de longueur 2 sont maintenant connus.

À la date T = 1 minute, chaque routeur envoie à nouveau sa table à ses voisins. Sans reprendre le détail des opérations, il est aisé de voir que les tables seront mises à jour avec les chemins de longueur 3. Si on traite les opérations comme précédemment dans l'ordre alphabétique, les tables sont :

routage(A) = [(A, 0, –) ; (B, 1, B) ; (C, 2, B) ; (D, 3, B) ; *(E, 1, E) ;* (F, 3, B)*].*

routage(B) = [(A, 1, A) ; (B, 0, –) ; (C, 1, C) ; (D, 2, C) ; (E, 1, E) ; (F, 2, C)].

routage(C) = [(A, 2, B) ; (B, 1, B) ; (C, 0, –) ; (D, 1, D) ; (E, 1, E) ; (F, 1, F)].

routage(D) = [(A, 3, C) ; *(B, 2, C) ; (C, 1, C) ; (D, 0, –) ; (E, 2, C) ; (F, 1, F)].*

routage(E) = [(A, 1, A) ; (B, 1, B) ; (C, 1, C) ; (D, 2, C) ; (E, 0, –) ; (F, 1, F)].

routage(F) = [(A, 2, E) ; (B, 2, C) ; (C, 1, C) ; (D, 1, D) ; (E, 1, E) ; (F, 0, –)].

À la date T = 1,5 minute, chaque routeur envoie à nouveau sa table à ses voisins et il n'y a aucune mise à jour : l'algorithme a convergé.

Remarque

La connaissance qu'ont les différents routeurs se borne à leurs voisins immédiats. Ainsi pour *A* les destinations *B, C, D* et *F* sont accessibles en passant par *B* et la destination *E* en passant par *E*.

b La liaison *CE* tombe en panne, on se trouve avec les tables suivantes :

routage(A) = [(A, 0, –) ; (B, 1, B) ; (C, 2, B) ; (D, 3, B) ; (E, 1, E) ; (F, 3, B)].

routage(B) = [(A, 1, A) ; (B, 0, –) ; (C, 16, C) ; (D, 16, C) ; *(E, 1, E) ;* (F, 16, C)*].*

routage(C) = [(A, 16, B) ; (B, 16, B) ; *(C, 0, –) ; (D, 1, D) ; (E, 1, E) ; (F, 1, F)].*

routage(D) = [(A, 3, C) ; (B, 2, C) ; (C, 1, C) ; (D, 0, –) ; (E, 2, C) ;(F, 1, F)].

routage(E) = [(A, 1, A) ; (B, 1, B) ; (C, 1, C) ; (D, 2, C) ; (E, 0, –) ; (F, 1, F)].

routage(F) = [(A, 2, E) ; (B, 2, C) ; (C, 1, C) ; (D, 1, D) ; (E, 1, E) ; (F, 0, –)].

B et *C* signalent à leurs voisins les destinations injoignables. Et donc ceux-ci modifient leurs tables de routage :

routage(A) = [(A, 0, –) ; (B, 1, B) ; (C, 16, B) ; (D, 16, B) ; *(E, 1, E) ;* (F, 16, B)*].*

routage(D) = [(A, 16, C) ; (B, 16, C) ; *(C, 1, C) ; (D, 0, –) ; (E, 2, C) ;(F, 1, F)].*

routage(E) = [(A, 1, A) ; (B, 1, B) ; (C, 1, C) ; (D, 2, C) ; (E, 0, –) ; (F, 1, F)].

routage(F) = [(A, 2, E) ; (B, 16, C) ; *(C, 1, C) ; (D, 1, D) ; (E, 1, E) ; (F, 0, –)].*

On constate que le routeur *E* n'est pas concerné.

À la prochaine échéance de mise à jour régulière (intervalle de 30 secondes), la diffusion des nouvelles tables va donner lieu à de multiples mises à jour. En particulier, quand *E* transmet sa table, les autres routeurs apprennent l'existence des chemins *EB* et *EC* qui vont se substituer au chemin *BC* défaillant.

routage(A) = [(A, 0, –) ; (B, 1, B) ; (C, 2, E) ; (D, 3, E) ; *(E, 1, E) ;* (F, 2, E)*].*

routage(B) = [(A, 1, A) ; (B, 0, –) ; (C, 2, E) ; (D, 3, C) ; *(E, 1, E) ;* (F, 2, E)*].*

routage(C) = [(A, 2, E) ; (B, 2, E) ; *(C, 0, –) ; (D, 1, D) ; (E, 1, E) ; (F, 1, F)].*

routage(D) = [(A, 16, C) ; (B, 16, C) ; (C, 1, C) ; (D, 0, –) ; (E, 2, C) ; (F, 1, F)].

routage(E) = [(A, 1, A) ; (B, 1, B) ; (C, 1, C) ; (D, 2, C) ; (E, 0, –) ; (F, 1, F)].

routage(F) = [(A, 2, E) ; (B, 2, E) ; (C, 1, C) ; (D, 1, D) ; (E, 1, E) ; (F, 0, –)].

À l'échéance suivante, l'information parvient à *D* et les tables sont alors :

routage(A) = [(A, 0, –) ; (B, 1, B) ; (C, 2, E) ; (D, 3, E) ; (E, 1, E) ; (F, 2, E)].

routage(B) = [(A, 1, A) ; (B, 0, –) ; (C, 2, E) ; (D, 3, C) ; (E, 1, E) ; (F, 2, E)].

routage(C) = [(A, 2, E) ; (B, 2, E) ; (C, 0, –) ; (D, 1, D) ; (E, 1, E) ; (F, 1, F)].

routage(D) = [(A, 3, C) ; (B, 3, C) ; (C, 1, C) ; (D, 0, –) ; (E, 2, C) ; (F, 1, F)].

routage(E) = [(A, 1, A) ; (B, 1, B) ; (C, 1, C) ; (D, 2, C) ; (E, 0, –) ; (F, 1, F)].

routage(F) = [(A, 2, E) ; (B, 2, E) ; (C, 1, C) ; (D, 1, D) ; (E, 1, E) ; (F, 0, –)].

À l'échéance suivante, il n'y a plus de changement, l'algorithme a à nouveau convergé.

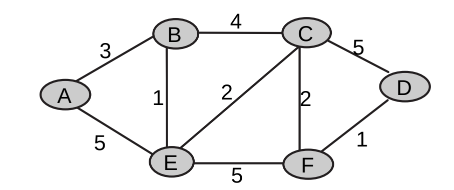
EXERCICE 3 ROUTAGE AVEC OSPF

Énoncé On considère le même système que dans l'exercice 1 et on applique cette fois-ci l'algorithme OSPF, sachant que les coûts de liaison ont été mesurés et sont illustrés à la figure 8.7.

AB = 3, *AE* = 5, *BC* = 4, *BE* = 1, *CD* = 5, *CE* = 2, *CF* =2, *DF* = 1, *EF* = 5.

Figure 8.7

Système autonome avec mesure de l'état des liens.



a Établissez la table de routage du nœud *A*, considéré comme routeur désigné du système.

b Le lien entre *C* et *E* tombe en panne. Les routeurs *C* et *E* diffusent l'information dans le système. Montrez comment *A* met à jour sa table de routage.

Solution **a** L'état initial est le suivant :

états(A) = [(B, 3, B) ; (E, 5, E)].

états(B) = [(A, 3, A) ; (C, 4, C) ; (E, 1, E)].

états(C) = [(B, 4, B) ; (D, 5, D) ; (E, 2, E) ; (F, 2, F)].

états(D) = [(C, 5, C) ; (F, 1, F)].

états(E) = [(A, 5, A) ; (B, 1, B) ; (C, 2, C) ; (F, 5, F)].

états(F) = [(C, 2, C) ; (D, 1, D) ; (E, 5, E)].

Les routeurs du système autonome effectuent une diffusion (contrôlée) de leurs informations à l'intérieur du système autonome et chacun peut reconstituer la cartographie du système (voir tableau 8.1).

Tableau 8.1

Matrice des états des liens du système autonome

État	A	B	C	D	E	F
A	0	3	x	x	5	x
B	3	0	4	x	1	x
C	x	4	0	5	2	2
D	x	x	5	0	x	1
E	5	1	2	x	0	5
F	x	x	2	1	5	0

Dans cette matrice symétrique, il y a des zéros sur la diagonale (coût nul) et des x pour les liaisons qui n'existent pas.

Calculons la table de routage du routeur *A*.

A connaît deux routes *AB* de coût 3 en passant par *B* directement et *AE* de coût 5 en passant par *E* directement. Il place la route *AE* en attente (5 est moins bon que 3) et valide la route *AB* à partir de laquelle il explore les chemins passant par *B*.

A	AB,3,B	x	x	*AE*, 5, *E*	x

La route *BC* est de coût 4 donc *AC* sera de coût 3 + 4 = 7 en passant par *B*. La destination *C* n'était pas encore connue, la route est ajoutée dans la table.

La route *BE* est de coût 1 donc *AE* sera de coût 3 + 1 = 4 en passant par *B* ce qui est meilleur que la route actuellement connue, la table est mise à jour.

A	AB,3,B	*AC*, 7, *B*	x	AE,4,B	x

Le routeur *A* place la route *AC* en attente (7 est moins bon que 4) et valide maintenant la route *AE* à partir de laquelle il explore les chemins passant par *E*(*).

Les routes *EA* et *EB* n'apportent rien, ni nouvelle destination, ni route meilleure que celles déjà connues.

La route *EC* est de coût 2 donc *AC* sera de coût 4 + 2 = 6 en passant par *B* ce qui est meilleur que la route actuellement connue, la table est mise à jour.

La route *EF* est de coût 5 donc *AF* sera de coût 4 + 5 = 9 en passant par *E*. La destination *F* n'était pas encore connue, la route est ajoutée dans la table.

A	AB,3,B	AC,6,E	x	AE,4,B	*AF*, 9, *E*

Le routeur *A* place la route *AF* en attente (9 est moins bon que 6) et valide maintenant la route *AC* à partir de laquelle il explore les chemins passant par *C*.

Les routes *CB* et *CE* n'apportent rien, ni nouvelle destination, ni route meilleure que celles déjà connues.

La route *CD* est de coût 5 donc *AD* sera de coût 6 + 5 = 11 en passant par *C*. La destination *D* n'était pas encore connue, la route est ajoutée dans la table.



La route *CF* est de coût 2 donc *AF* sera de coût 6 + 2 = 8 en passant par *C* ce qui est meilleur que la route actuellement connue, la table est mise à jour.

A	AB,3,B	AC,6,E	*AD*, 11, *C*	AE,4,B	AF,8,C

Le routeur *A* place la route *AD* en attente (11 est moins bon que 8) et valide maintenant la route *AF* à partir de laquelle il explore les chemins passant par *F*.

Les routes *FC* et *FE* n'apportent rien.

La route *FD* est de coût 1 donc *AD* sera de coût 8 + 1 = 9 en passant par *F* ce qui est meilleur que la route actuellement connue, la table est mise à jour.

A	AB,3,B	AC,6,E	*AD*, 9, *F*	AE,4,B	AF,8,C

La route *AD* est validée à son tour et les routes *DC* et *DE* n'apportent rien.

L'algorithme est terminé.

La table de routage de *A* est alors la suivante :

routage(*A*) = [(*A*, 0, –) ; (*B*, 3, *B*) ; (*C*, 6, *B*) ; (*D*, 9, *B*) ; (*E*, 4, *B*) ; (*F*, 8, *B*)].

Toutes les routes passent par le routeur *B* comme l'illustre la figure 8.8.

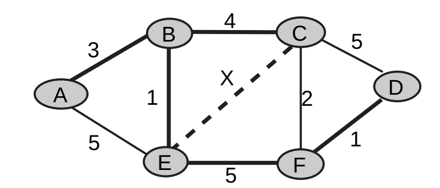


Figure 8.8

Système autonome avec les meilleures routes en gras.

b Le lien entre *C* et *E* tombe en panne. Les routeurs *C* et *E* diffusent l'information dans le système. La matrice du système est maintenant illustrée au tableau 8.2.

Tableau 8.2

Matrice des états des liens du système autonome après la panne E-C

État	A	B	C	D	E	F
A	0	3	x	x	5	x
B	3	0	4	x	1	x
C	x	4	0	5	x	2
D	x	x	5	0	x	1
E	5	1	x	x	0	5
F	x	x	2	1	5	0

L'application de l'algorithme par le routeur *A* est la même que précédemment jusqu'à l'étape marquée (*) dans la correction de la question a et que nous reprenons ci-après.

A	AB,3,B	*AC*, 7, *B*	x	AE,4,B	x

Le routeur *A* place la route *AC* en attente (7 est moins bon que 4) et valide maintenant la route *AE* à partir de laquelle il explore les chemins passant par *E*(*).

Les routes *EA* et *EB* n'apportent rien. La route *EF* est de coût 5 donc *AF* sera de coût 4 + 5 = 9 en passant par *E*. La destination *F* est ajoutée.

A	AB,3,B	*AC*, 7, *B*	x	AE,4,B	*AF*, 9, *E*

Le routeur A place la route AF en attente et valide la route AC à partir de laquelle il explore les chemins passant par *C*. On obtient :

A	AB,3,B	AC,7,B	*AD*, 12, *C*	AE,4,B	*AF*, 9, *E*

et enfin :

A	AB,3,B	AC,7,B	AD,10,F	AE,4,B	AF,9,E

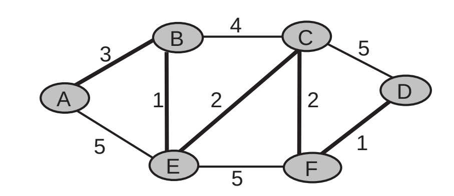
La nouvelle table de routage de A est maintenant :

$$\text{routage}(A) = [(A, 0, -) ; (B, 3, B) ; (C, 7, B) ; (D, 10, E) ; (E, 4, B) ; (F, 9, E)].$$

Les routes retenues sont illustrées à la figure 8.9.

Figure 8.9

Routes recalculées après la panne.





Les applications

De nombreux applicatifs largement répandus utilisent TCP/IP. Nous en aborderons quelques-uns dans ce chapitre : le service de configuration dynamique des machines, le service de noms de domaine, le courrier électronique, le transfert de fichiers, sans oublier l'incontournable navigation sur le Web. Les deux premières applications sont, en fait, des applications internes, utiles au bon fonctionnement des réseaux. Elles rendent des services indispensables comme la distribution des adresses IP et la correspondance entre les noms symboliques des machines et leurs adresses IP. Les autres concernent directement les utilisateurs et leurs besoins de communication à travers les réseaux.
Le courrier électronique (ou messagerie électronique) est l'une des premières applications développées dans Internet. La navigation sur le Web est à l'origine de l'engouement populaire pour Internet.

1 Service de configuration dynamique DHCP (*Dynamic Host Configuration Protocol*)

Le service de configuration dynamique DHCP fournit un cadre pour transmettre des informations de configuration à des machines d'un réseau TCP/IP : il leur communique leurs paramètres de configuration, en particulier leurs adresses IP. C'est un service réseau dont nous donnerons un bref aperçu du principe de fonctionnement.

1.1 Principe de fonctionnement

La RFC 2131 décrit le service de configuration dynamique. Celui-ci est construit sur un modèle client/serveur et utilise les ports 67 et 68. Il comprend une méthode d'attribution d'adresse IP à la machine cliente et utilise un protocole DHCP pour transmettre l'ensemble des paramètres de configuration. Après obtention des paramètres fournis par un serveur DHCP, le client est capable de communiquer avec n'importe quel autre utilisateur d'Internet.

Le service DHCP garantit l'unicité d'une adresse IP dans le réseau. C'est une aide précieuse pour l'administrateur du réseau puisqu'il permet de supprimer la configuration manuelle des machines du réseau (celle-ci reste toujours possible et doit être compatible avec l'allocation dynamique). En outre, il évite les erreurs liées à cette configuration manuelle. Il sera très utile pour le déploiement d'un grand parc de machines qui possèdent les mêmes caractéristiques et dont la seule différence est la configuration réseau.

Le serveur DHCP dispose d'une plage d'adresses IP à attribuer. Quand un client en demande une pour la première fois, le serveur lui en fournit une qu'il n'a pas encore utilisée. S'il a déjà distribué toutes ses adresses IP, il réutilise celle d'un client qui n'est plus connecté au réseau. Quand un client a déjà obtenu dynamiquement une adresse IP lors d'une connexion antérieure, le serveur lui fournit *a priori* la même adresse, si c'est possible. La réattribution d'une adresse ne s'avère nécessaire qu'en cas de pénurie. En effet, DHCP peut s'utiliser avec un nombre de clients supérieur au nombre d'adresses IP disponibles. Cette méthode ne convient pas pour définir l'adressage des machines ayant un rôle de serveurs, qui doivent avoir une adresse fixe. On utilise donc également l'attribution fixe d'une adresse IP pour les machines qui en ont besoin. Dans ce cas, l'adresse MAC qui identifie la machine est mise en correspondance avec une adresse IP fixe. Par ailleurs, l'attribution fixe est une façon de lutter contre les intrusions dans le réseau !

Il peut y avoir plusieurs serveurs DHCP actifs simultanément dans le réseau (pour assurer une redondance du service et optimiser les performances d'accès). À l'inverse, il n'est pas indispensable d'en avoir un pour chaque sous-réseau. DHCP peut fonctionner à travers des routeurs, en utilisant des relais. C'est un protocole construit comme une extension de BOOTP (*Bootstrap Protocol*), protocole de démarrage des stations sans disque. Il apporte en complément la gestion dynamique des adresses IP et contient un grand nombre d'options. De plus, la machine est directement opérationnelle après avoir reçu ses paramètres de configuration et n'a pas besoin de redémarrer.

1.2 Échanges DHCP

Les messages du protocole DHCP utilisent UDP au niveau transport. Ce dernier est suffisant pour des échanges simples sur le réseau local : le protocole DHCP fonctionne en mode non connecté puisque les interlocuteurs ne se connaissent pas à l'avance.



Le protocole contient plusieurs messages : *DHCPDiscover*, *DHCPOffer*, *DHCPRequest*, *DHCPAck*, *DHCPDecline* et *DHCPRelease*.

Par définition, une machine client qui cherche à obtenir une adresse IP n'a aucune information de configuration réseau. Elle sait seulement qu'elle doit utiliser DHCP (*a priori*, elle ne connaît même pas l'adresse du serveur DHCP à utiliser). Elle diffuse un message baptisé *DHCPDiscover*, encapsulé dans un datagramme UDP, sur le réseau local où elle se trouve. Dans le cas d'un réseau utilisant des VLAN, le commutateur assure la diffusion au VLAN auquel le client est rattaché. Il est possible que d'autres machines du même VLAN soient accessibles au-delà du routeur. Un agent relais est alors nécessaire pour assurer la transmission au serveur DHCP du réseau.

Le serveur répond avec un message *DHCPOffer* qui contient l'adresse IP proposée au client. Celui-ci peut recevoir autant de messages *DHCPOffer* que de serveurs DHCP actifs. Au bout d'un certain temps, le client renouvelle sa requête *DHCPDiscover* s'il ne reçoit pas au moins une réponse. S'il a reçu plusieurs réponses, il en choisit une et diffuse un message *DHCPRequest* qui indique le serveur choisi et l'adresse IP proposée.

Les serveurs non sélectionnés invalident leur proposition. Le serveur choisi vérifie la disponibilité de l'adresse proposée (en utilisant ICMP *Echo Request* ou *ping*). S'il n'obtient pas de réponse à son *ping*, l'adresse est *a priori* disponible. Il envoie alors la configuration complète au client dans un message *DHCPAck*.

Le client effectue lui aussi un dernier test (avec le protocole ARP). Il cherche, avec une requête ARP, l'adresse MAC de celui qui possède l'adresse IP proposée. Il ne doit obtenir aucune réponse puisque l'adresse IP n'est pas encore distribuée. En cas de problème, il peut refuser la configuration (message *DHCPDecline*) et recommencer le processus. Enfin, un client quittant le réseau abandonne son adresse par un message *DHCPRelease*.

Les contrôles du serveur puis du client servent à garantir l'unicité de l'adresse IP dans le réseau. La présence de plusieurs serveurs DHCP susceptibles de répondre complique la tâche. On peut aussi imaginer une machine qui usurperait le rôle de serveur DHCP avec des intentions malveillantes…

Pour des raisons d'optimisation des ressources réseau, les serveurs DHCP ne délivrent les adresses IP que pour une durée limitée appelée *bail*. Un client dont le bail arrive à terme peut en demander le renouvellement par le message *DHCPRequest*. De même, lorsque le serveur voit un bail arriver à terme, il propose au client de prolonger son bail, avec la même adresse IP. S'il ne reçoit pas de réponse valide dans un délai fixé, il récupère l'adresse IP concernée. Quand les utilisateurs éteignent et rallument très souvent leurs machines, il est intéressant de travailler avec des baux de courte durée.

2 Service d'annuaire

Dans les années 1970, il n'y avait que quelques machines interconnectées. L'intégration d'une nouvelle machine sur le réseau nécessitait la mise à jour d'un fichier de configuration qui contenait une table de correspondance entre noms de machines (gérées par les humains) et adresses IP (utilisées dans le réseau). Ce procédé est devenu rapidement obsolète pour Internet.

Il faut toujours connaître l'adresse IP de la destination. L'être humain ne connaît la destination que par son nom symbolique, par exemple *machine.societe.pays* ou pour la messagerie *prenom.nom@fournisseur.organisme*. Ce nom symbolique est facile à comprendre et à retenir.

Le système mis en place, pour répondre au problème posé, est le DNS (*Domain Name System*). Les premières RFC (882 et 883) définissant ce service datent de 1984. Un service d'annuaire fait la correspondance entre le nom symbolique et l'adresse IP. Notons qu'un nom symbolique peut correspondre à plusieurs adresses IP différentes (pour faire de l'équilibrage de charge par exemple) et réciproquement plusieurs noms symboliques peuvent correspondre à une seule adresse IP (une machine qui héberge plusieurs applications).

Le fonctionnement automatique du service d'annuaire, à la différence du service d'annuaire téléphonique, simplifie grandement l'accès aux ressources réseau. Il a, en partie, provoqué le grand succès du Web. En effet, l'application cliente émet automatiquement la requête d'interrogation de l'annuaire dès qu'elle doit faire la correspondance entre un nom symbolique et une adresse IP. Elle attend la réponse. Une fois qu'elle l'a obtenue, elle est capable d'envoyer des messages au destinataire, désormais connu par son adresse IP. Le temps passé à cette requête/réponse est tellement bref que l'utilisateur ne s'aperçoit de rien.

Comme dans tous les systèmes distribués, le serveur DNS interrogé peut ne pas connaître la réponse à la requête reçue. En revanche, il connaît l'adresse d'un autre serveur qui sait répondre. Les serveurs DNS sont ainsi organisés en un ensemble collaboratif réparti à l'échelle de la planète. Ils collaborent de deux façons : soit le serveur renvoie au client la référence du serveur qui sait répondre (mode itératif), soit il poursuit lui-même la recherche auprès de ce serveur et fournit au client la réponse attendue (mode récursif). Le mode itératif est obligatoire, le mode récursif est une option.

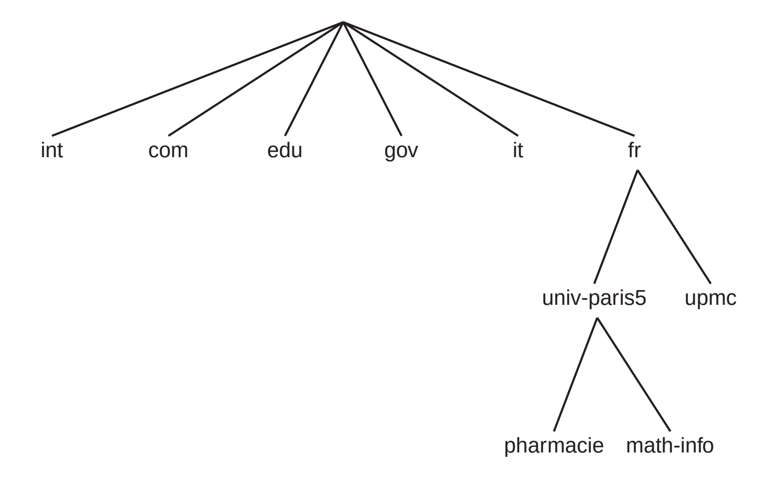
2.1 Espace des noms

Arbre de nommage

L'espace des noms symboliques (ou noms *canoniques*) définit un ensemble de règles de nommage et les données associées. L'ensemble est organisé sous la forme d'un arbre, dont chaque nœud et chaque feuille contiennent un certain nombre d'informations (voir figure 9.1). Les requêtes d'interrogation servent à obtenir ces différentes informations. Ainsi, pour accéder à une information donnée, on précise dans la requête le nom de domaine concerné et le type d'information désirée.

Figure 9.1

Arbre de nommage.



À la racine de l'arbre se trouve un point. Le premier niveau (*top level domain*) correspond aux suffixes bien connus à trois lettres : *com*, *net*, *int*… et aux différents pays qui n'ont qu'un suffixe à deux lettres : *fr* pour France, *it* pour Italie, *de* pour Allemagne… (norme ISO 3166). Les domaines *com* et *net* étaient initialement dévolus aux entreprises commerciales ou



concernées par les réseaux. Ils sont maintenant accessibles à tout un chacun, y compris aux particuliers. Aux États-Unis, *edu* correspond aux établissements scolaires ; *gov* aux organismes gouvernementaux. En France, un découpage similaire existe avec le sous-domaine *gouv.fr* pour les institutions relevant de l'État.

Délégation

L'administration des noms de domaine possède plusieurs niveaux hiérarchiques : l'ICANN (*Internet Corporation for Assigned Names and Numbers*) aux États-Unis est responsable de la coordination mondiale. Cet organisme a délégué à RIPE-NCC (*Réseaux IP Européens*[1] *– Network Coordination Center*) la gestion des noms de domaine en Europe (et à d'autres organismes similaires dans les autres continents). Le domaine représente la sous-arborescence à partir d'un nœud donné, c'est-à-dire l'ensemble des informations de tous les nœuds qui ont celui-ci comme racine. La définition récursive des arbres fait qu'un sous-domaine est un domaine[2].

RIPE-NCC a délégué à l'AFNIC (*Association française pour le nommage Internet en coopération*) la gestion des noms de domaine en France. L'AFNIC enregistre tous les noms de sous-domaine du domaine .fr avec un gérant pour chaque domaine (par exemple *univ-paris5.fr* est géré par l'université René Descartes, également appelée Paris 5). Le gérant du domaine *domaine.fr* est responsable de la délégation des noms de domaine de la forme *sous-domaine.domaine.fr* et de la désignation d'un administrateur de chaque sous-domaine.

Nom absolu, nom relatif et alias

Chaque machine porte un nom unique, mais on peut utiliser des noms plus génériques qui sont des *alias*. Le nom complet d'une machine sur Internet se compose du nom de la machine dans le réseau local suivi du nom de la zone à laquelle elle appartient. On parle du FQDN de la machine (*Fully Qualified Domain Name*). Au sein du réseau local, on se contente d'un adressage relatif en n'utilisant que le nom de la machine. Dans ce cas, les logiciels de communication utilisés (et correctement configurés) rajoutent automatiquement le nom de zone pour construire le FQDN.

Exemple

La machine *toutatis* héberge un serveur Web et un serveur de messagerie. Son nom symbolique est *toutatis.societe.pays*. Mais l'administrateur a défini deux alias pour cette machine : *www.societe.pays* et *mailhost.societe.pays*. Vu de l'extérieur de la société, il y a un serveur Web et un serveur de messagerie, peu importe la machine réelle qui les héberge. De plus, le changement de cette machine réelle (la messagerie passe sur une nouvelle machine dénommée en interne *bélénos*) n'a pas d'impact sur la configuration des clients de messagerie.

2.2 Serveurs de noms

Les serveurs de noms, ou serveurs DNS, sont des machines qui assurent la traduction des noms en adresses IP et réciproquement. On parle de *résolution* d'un nom ou d'une adresse. Pour cela, ils possèdent des informations sur l'architecture de l'arbre et sur les données associées. *A priori*, un serveur de nom peut mémoriser des informations concernant n'importe quelle partie de l'arbre. En général, il contient des informations complètes sur un sous-arbre, ainsi que des références à d'autres serveurs susceptibles de fournir des informations sur le reste de l'arbre. Lorsqu'un serveur a la connaissance complète d'un

1. Remarquez l'un des rares acronymes en français !

2. Un arbre possède une racine et plusieurs branches avec des nœuds d'où partent de nouvelles branches et ainsi de suite. Si on coupe une branche à la hauteur d'un nœud, l'ensemble coupé est un nouvel arbre dont le nœud de la coupure est la racine.

sous-arbre (ou *zone*), on dit qu'il est le serveur *officiel* de cette zone. Pour des raisons d'efficacité, un serveur peut utiliser un mécanisme de cache pour stocker des informations relatives à d'autres zones pour lesquelles il n'est pas le serveur officiel.

Une zone est accessible par l'intermédiaire d'un ou plusieurs serveurs DNS. La présence de plusieurs serveurs garantit la continuité du service, même en cas de panne de l'un d'eux. En fait, il existe trois types de serveurs de noms : *primaires*, *secondaires* et *cache*. Les serveurs primaires gèrent la mise à jour des correspondances entre nom symbolique et adresse IP. Les serveurs secondaires récupèrent une copie des enregistrements d'un serveur primaire (pour décharger ce dernier ou le suppléer en cas d'arrêt). Les serveurs cache ne disposent d'aucune base mais conservent les réponses des requêtes résolues par des serveurs primaires ou secondaires.

Dans sa zone, un serveur DNS officiel connaît les adresses des serveurs de ses sous-domaines (s'il y a délégation) et celles des serveurs de la zone supérieure. Ainsi, un serveur interrogé sur la résolution d'un nom portant sur un sous-domaine qu'il gère peut répondre directement s'il a l'information, soit fournir la liste des serveurs de noms susceptibles de répondre. S'il est interrogé sur la résolution d'un nom d'un domaine qu'il ne gère pas, il contacte le serveur de noms de sa zone supérieure. Un tel système est efficace car les serveurs intermédiaires conservent en cache les requêtes déjà résolues. La zone supérieure racine connaît les serveurs DNS racine (*root-servers*). Il s'agit de treize machines dont dix sont aux États-Unis (cinq sur la côte est, cinq sur la côte ouest), deux en Europe et une au Japon.

Les informations stockées par le serveur constituent un fichier appelé *db* et dont les enregistrements sont nommés RR (*ressource records*). Le fichier *db* est facile à lire. Il contient non seulement l'adresse IP (enregistrement *A* pour IPv4 et *AAAA* pour IPv6), le nom canonique (enregistrement *CNAME*) et les alias, mais aussi de multiples informations comme :

- le serveur de messagerie associé au domaine (enregistrement MX, *Mail eXchange server*) ;
- le pointeur entre l'adresse IP et le nom canonique (enregistrement PTR, *pointer*), qui sert à la résolution inverse ;
- le statut officiel (enregistrement SOA, *Start of Authority*), qui détermine le serveur DNS officiel primaire de la zone avec l'ensemble de ses paramètres ;
- les autres serveurs de noms associés au domaine (enregistrements NS, *Name Server*).

L'enregistrement SOA contient plusieurs temporisateurs (exprimés en secondes) et définissant les temps de rafraîchissement, d'attente, d'échéance et de vie :

- Le temps de rafraîchissement (*Refresh*) est l'intervalle de temps entre deux vérifications d'un serveur secondaire sur le serveur primaire pour savoir s'il y a eu des modifications et si une mise à jour est nécessaire.
- Le temps d'attente (*Retry*) d'un serveur secondaire avant de renouveler sa mise à jour si la précédente a échoué.
- Le temps d'échéance (*Expire*) au bout duquel le serveur efface les informations qui n'ont pas pu être mises à jour.
- Le temps de vie (*Time To Live* ou *Minimum*) est la durée pendant laquelle un serveur DNS peut conserver en cache un enregistrement du fichier de la base de données.

Ces différents temporisateurs sont configurés pour optimiser le trafic entre les serveurs : la mise en cache réduit les délais d'obtention d'une information. Encore faut-il que celle-ci soit correcte, d'où le besoin de rafraîchir régulièrement les données.

Le bon fonctionnement du service DNS est vital pour tous les réseaux. Les coupures de courant, les ruptures de connexion pourraient être préjudiciables au service. Quand un

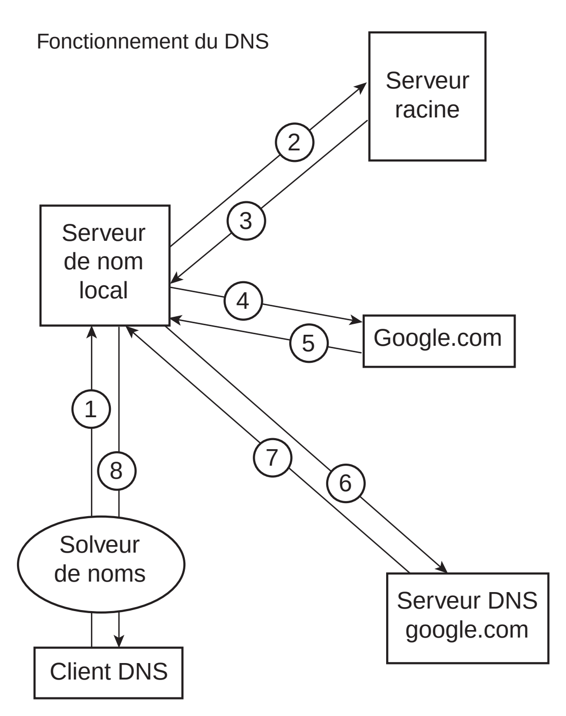


réseau est coupé du reste du monde, ses serveurs de noms peuvent se trouver dans des situations délicates : ils n'ont plus accès aux serveurs du domaine parent ni aux serveurs racine. C'est pourquoi on utilise des serveurs multiples et redondants, des retransmissions de requêtes et des tentatives répétées de chargement des informations de zone, en exploitant les différents temporisateurs prévus à cet effet.

2.3 Clients du service de noms de domaines : les solveurs

Les solveurs de noms (*resolvers*) sont les programmes qui, à la suite d'une demande provenant d'une application, obtiennent l'information recherchée auprès des serveurs de noms en les interrogeant. Ils utilisent l'information reçue pour répondre à l'application. Le solveur est donc un processus client, directement accessible par les programmes utilisateur. La figure 9.2 illustre le rôle de chacun.

Figure 9.2
Échanges DNS.



1 Envoi de la requête sur le serveur de nom local.
2 Si le serveur local ne sait pas répondre, il interroge le serveur de noms racine.
3 Envoi de l'adresse IP du serveur qui gère la zone.com.
4 Interrogation du serveur ayant autorité sur la zone.com.
5 Envoi de l'adresse IP du serveur DNS de google.com.
6 Interrogation du serveur DNS google.com.
7 Envoi de l'adresse IP de www.google.com
8 Le serveur de nom local envoie l'adresse IP à son client DNS.

L'application cliente sollicite le solveur de nom qui répond immédiatement (informations mises en cache dans le fichier /etc/hosts, par exemple) ou s'adresse au serveur de noms qu'il connaît (information de configuration de la machine) pour lui soumettre la requête

de l'utilisateur. Le solveur est un client vis-à-vis du serveur de noms. Utiliser un solveur décharge le programme sur lequel travaille l'utilisateur et minimise le trafic sur le réseau.

Remarque

L'utilisation de la commande `ipconfig /all` pour les postes sous Windows fournit (entre autres) l'information sur les serveurs DNS. Ci-après la réponse pour un poste connecté en ADSL chez Wanadoo :

```
serveurs DNS  80.10.246.1
              80.10.246.132
```

Le solveur de la machine interroge en priorité le premier serveur DNS 80.10.246.1 et, si celui-ci ne répond pas, le second 80.10.246.132. Ces deux adresses ont été fournies par le fournisseur d'accès à Internet.
La commande nslookup permet de vérifier le rôle des deux serveurs. Elle affiche :

```
serveur par défautdns-adsl-gpe1-a.wanadoo.fr
address       80.10.246.1.
```

2.4 Dialogue avec le serveur DNS

Les requêtes envoyées aux serveurs de noms en vue de consulter les données stockées utilisent UDP au niveau transport, suffisant pour un dialogue requête/réponse. Le port 53 est réservé au dialogue avec le serveur DNS, dont la réaction peut être de trois types : elle répond à la question posée, renvoie vers un autre serveur ou signale une erreur.

Un protocole standardisé définit le format des requêtes et des réponses DNS. Celles-ci possèdent un en-tête contenant des champs fixes toujours présents et quatre sections décrivant la demande et les réponses obtenues. Parmi les champs fixes, on trouve 4 bits appelés *code d'opération*. Le code d'opération donne des informations sur la nature du message (requête, réponse…). Les quatre sections sont *Question*, *Answer*, *Authority*, *Additional*.

La section *Question* contient la question (nom d'hôte ou de domaine sur lequel on cherche des renseignements et type de renseignements recherchés). La section *Answer* contient les enregistrements qui répondent à la question. *Authority* indique les serveurs officiels, ceux qui ont une connaissance complète de la zone considérée. Enfin *Additional* contient des enregistrements supplémentaires pour exploiter les informations contenues dans les autres sections.

2.5 Résolution de noms inverse

Le domaine `in addr.arpa` sert à la résolution inverse : il retrouve le nom d'une machine à partir de son adresse IP. Les nœuds de ce domaine correspondent aux différents numéros (de 0 à 255) de l'adresse IP, pris dans le sens inverse de la lecture classique. Ainsi l'adresse IP 123.45.67.89 correspond à une machine appelée *mon_ordi.mon_domaine.mon_pays*. La branche de l'arbre correspondant part de la racine puis va sur le nœud arpa puis i-n-addr puis 123 puis 45 puis 67 et enfin 89. L'information placée à cet endroit, soit *89.67.45.123.in-addr.arpa*, renvoie vers le nom qualifié *mon_ordi.mon_domaine.mon_pays*, comme le montre la figure 9.3. On comprend mieux pourquoi les adresses IP sont exploitées à l'envers : le domaine 123.in-addr.arpa peut être délégué aux administrateurs du réseau 123.0.0.0.

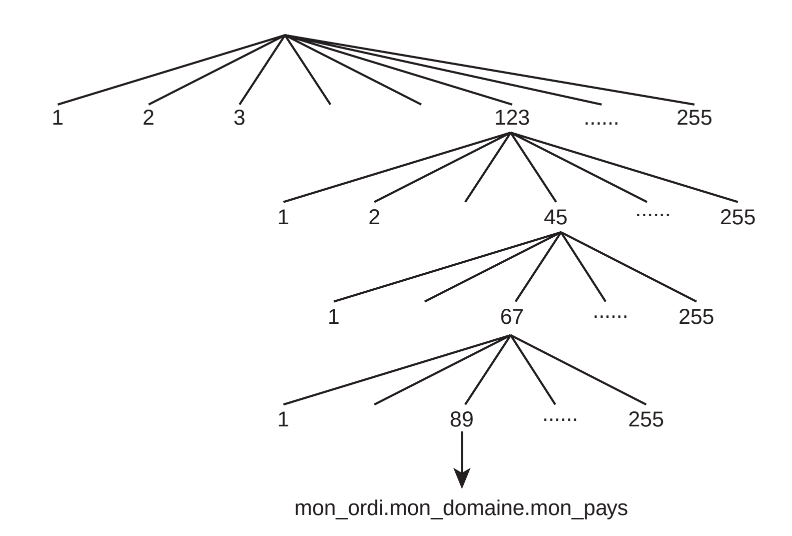


Figure 9.3
Résolution inverse.

2.6 Annuaires électroniques d'entreprise

Les annuaires électroniques sont des bases de données spécialisées, qui stockent des informations de manière hiérarchique et permettent ensuite de les rechercher rapidement pour les exploiter. Les annuaires contiennent, outre l'enregistrement de l'ensemble des utilisateurs (avec des données correspondant à leurs fonctions dans l'entreprise), celui des machines et des applications. L'objectif principal d'un annuaire est d'assurer l'authentification[3] des utilisateurs grâce à un mot de passe et de définir leurs droits vis-à-vis des différentes applications déployées dans l'entreprise.

On interroge très souvent un annuaire, on met plus rarement ses données à jour. Il est donc important de l'optimiser pour la recherche d'informations. Sa structure est hiérarchique, à l'instar de la structure de nommage vue précédemment. Dans les très grandes entreprises, il peut y avoir plusieurs serveurs d'annuaire, du fait de la grande masse d'informations à stocker. Ces différents serveurs doivent se synchroniser régulièrement pour fournir une information cohérente.

La norme ISO *X.500* a défini la notion de service d'annuaire. Elle accepte, à la différence du service DNS, plusieurs types de recherche, y compris des correspondances ou des informations incomplètes ; elle a, comme toute norme internationale, une vocation très générale. Elle repose sur une authentification des utilisateurs et plusieurs types de chiffrement des données. Les promoteurs d'Internet l'ont simplifiée et rebaptisée LDAP (*Lightweight Directory Access Protocol*). Sa version 3 (RFC 2251) tend aujourd'hui à devenir le standard d'accès aux annuaires.

Pour organiser les données dans un annuaire LDAP, quatre modèles de base existent : information, nommage, fonctionnel et sécurité. Le modèle d'information définit le type des informations contenues dans l'annuaire (pays, organisation, nom, prénom…). Le modèle de nommage définit les règles de dénomination des informations à partir d'un arbre. Le modèle fonctionnel définit l'accès et la mise à jour des informations. Enfin, le modèle de sécurité définit comment protéger les données et les accès.

La technologie SSO (*Single Sign On*) s'appuie sur l'annuaire et permet aux utilisateurs du réseau de l'entreprise d'accéder en toute transparence à l'ensemble des ressources autorisées, grâce à une authentification unique effectuée à l'accès initial dans le réseau. Avec un seul mot de passe, l'utilisateur accède aux applications auxquelles il a droit : l'annuaire

3. Ne pas confondre *authentification* (vérification de l'identité de l'utilisateur) et *autorisation* (vérification de ses droits d'accès) [voir les compléments pédagogiques, sur le site www.pearsoneducation.fr].

envoie directement à une application donnée le mot de passe nécessaire, sans que l'utilisateur soit obligé d'intervenir. Cette technologie améliore à la fois l'ergonomie d'accès aux applications et la sécurité du système d'information, tout en limitant la circulation des mots de passe.

3 Transfert de fichiers

Le *transfert de fichiers* est l'échange de longs documents entre ordinateurs (par opposition au courrier électronique, qui est plutôt destiné aux messages courts). FTP (*File Transfer Protocol*) et TFTP (*Trivial File Transfer Protocol*) sont deux protocoles de transfert de fichiers. Leur mode de fonctionnement est de type client/serveur. TFTP, comme son nom l'indique, possède des fonctionnalités réduites mais plus simples à gérer par rapport à FTP.

3.1 Serveurs et clients FTP (*File Transfer Protocol*)

Des milliers de serveurs sont connectés à Internet et proposent au public toutes sortes de logiciels à télécharger sur leurs propres machines. Mais les utilisateurs peuvent aussi, parfois, déposer leurs propres fichiers sur les serveurs. En anglais, on utilise les termes *download* pour le téléchargement de fichier dans le sens serveur vers client et *upload* pour le dépôt de fichier dans le sens client vers serveur. Nous utiliserons ici « télécharger » et « déposer ».

Pour les utilisateurs, plusieurs logiciels clients existent. Les premiers proposés sur le marché employaient une interface rudimentaire par ligne de commandes. Actuellement, une interface graphique simplifie les tâches. Cependant, même avec celle-ci, les logiciels clients sont souvent limités à la lecture et à l'écriture de fichiers, alors que les fonctionnalités de FTP autorisent la manipulation des fichiers à distance (lecture, écriture, mais aussi effacement, renommage…). Le transfert de fichiers assuré par FTP utilise les services de TCP. En effet, FTP est un protocole complexe qui nécessite une identification et une authentification de l'utilisateur par *login* et mot de passe. Un compte personnel sur un serveur permet d'y déposer des fichiers (des pages Web, par exemple). En pratique, tous les serveurs offrent un accès *anonyme*. Dans ce cas, le login de l'utilisateur est *anonymous.* La *Netiquette* recommande qu'on mette son adresse électronique comme mot de passe. L'accès anonyme ne permet que la lecture de fichiers appartenant à des répertoires dits « publics ». Tous les serveurs compressent les fichiers à télécharger pour limiter l'espace de stockage nécessaire et optimiser les temps de transfert vers l'utilisateur. Ce dernier doit alors disposer des utilitaires adaptés pour effectuer la décompression des fichiers importés sur sa machine.

3.2 Protocole FTP

Le protocole FTP utilise deux numéros de ports distincts pour le serveur. Il gère deux connexions TCP par session d'échange. Le port 21 sert à la connexion TCP qui supporte le dialogue d'établissement de connexion et d'authentification. Un serveur FTP peut travailler en *mode actif* ou en *mode passif*. En mode actif, le port 20 est utilisé pour la connexion TCP gérant le transfert des données du fichier. En mode passif, le serveur utilise un numéro de port quelconque, pour la connexion TCP gérant le transfert des données du fichier.

Lorsque le serveur FTP est en mode actif, il initie lui-même la deuxième connexion TCP avec le port 20. Si le client télécharge plusieurs fichiers, il y aura autant de connexions TCP sur le port 20 que de fichiers à transférer. Le numéro de port côté client progressant



d'une connexion à l'autre, il faut que le client apprenne au serveur le numéro du processus associé à chaque transfert de fichier. Le client FTP est dans ce cas serveur pour TCP : il fait une ouverture passive, dans l'attente de l'ouverture de connexion de la part du serveur. On comprend dans ce cas pourquoi on peut trouver des ouvertures passives de connexions TCP sur une machine qui, du point de vue de l'utilisateur, n'est qu'une machine cliente…

Remarque

Il faut comprendre que le rôle de client ou celui de serveur est indépendant du niveau dans l'architecture des protocoles. Ainsi une machine cliente pour FTP se retrouve être serveur pour TCP.

Si le client souhaite que le serveur travaille en mode passif, une commande FTP lui permet de faire la requête correspondante au serveur. Le serveur indique alors au client le numéro de port qui supportera la connexion TCP de transfert des données. Cette connexion de données reste à l'initiative du client.

Remarque

L'initiateur de la connexion TCP (c'est-à-dire celui qui envoie le premier message avec drapeau SYN à 1) est un paramètre important : on verra sur le site www.pearsoneducation.fr que la requête TCP de connexion avec drapeau SYN à 1 provenant de l'extérieur est souvent suspectée (tentative d'intrusion dans le réseau, par exemple) et de ce fait rejetée par l'équipement baptisé *pare-feu* qui protège le réseau de l'entreprise.

Dans les deux cas, FTP dispose d'un grand nombre de commandes permettant à l'utilisateur de naviguer dans l'arborescence des fichiers du serveur. Enfin, FTP transfère deux types de fichiers : ceux au format ASCII et ceux qui sont considérés comme de simples suites de données binaires. Le client doit configurer son application en indiquant le type de fichier, à moins que celle-ci ne sache reconnaître automatiquement son type (un fichier .txt est *a priori* codé en ASCII…). Un serveur FTP contient trois répertoires de base : le répertoire d'installation de l'application, le répertoire et tous les sous-répertoires nécessaires au stockage des données. Enfin, il utilise un répertoire pour les informations d'identification des usagers. De nombreuses failles de sécurité ont été constatées lors de l'échange de fichiers par FTP, sans parler des problèmes liés aux débordements de mémoire tampon. Nous abordons les problèmes de sécurité dans les compléments pédagogiques, sur le site www.pearsoneducation.fr.

3.3 Le protocole TFTP (*Trivial File Transfer Protocol*)

TFTP utilise pour sa part les services d'UDP. Il est limité au seul transfert de fichiers, sans authentification de l'utilisateur. De ce fait, un serveur TFTP n'offre que des possibilités d'accès à un nombre restreint de fichiers bien spécifiques. Il s'agit généralement des fichiers de démarrage de stations sans disque, et on l'associe dans ce cas au protocole BOOTP. Les messages du protocole TFTP se limitent à une *requête,* un message de *données,* un *accusé de réception* et un message d'*erreur.* La requête spécifie le nom du fichier ; les messages de données contiennent chacun un bloc – numéroté – de données appartenant au fichier désigné ; un accusé de réception spécifie le numéro du bloc acquitté ; et un message d'erreur signale un problème ce qui met fin au transfert. Le protocole sous-jacent étant UDP, la seule politique utilisable pour la gestion des messages de données est de type *Stop-and-Wait.*

4 La messagerie électronique

L'usage de la messagerie électronique varie selon qu'elle sert à diffuser de l'information (envoi unidirectionnel) ou à échanger des messages entre deux personnes ou plus (envoi multidirectionnel).

Dans l'entreprise, la diffusion possède un aspect relativement formel et contrôlé. Elle comprend des messages « pour information ». La diffusion s'appuie sur des listes d'adresses qui regroupent les personnes ayant un intérêt commun pour un type d'information donné. Pour éviter l'engorgement du réseau par la diffusion de messages volumineux vers un grand nombre de destinataires, on envoie par exemple un message réduit faisant mention d'un nouveau document accessible dans l'intranet par les destinataires du message.

Les échanges multidirectionnels de messagerie sont plutôt informels et concernent principalement la communication entre deux personnes avec éventuellement copie à quelques autres destinataires. Dans le cas d'un groupe, et si tout le monde a intérêt à connaître les réponses aux différentes questions posées, il est intéressant de constituer un *forum* et de stocker tous les échanges dans une base de données.

Dans le cadre des systèmes informatiques de messageries, on met en correspondance le nom d'une personne avec le nom de sa boîte aux lettres. Cette association s'effectue avec l'annuaire des personnes. Il faut ensuite localiser le système informatique de traitement chargé d'héberger la boîte aux lettres : cette relation est maintenue dans l'annuaire des machines. La préparation d'un message demande l'utilisation d'un traitement de texte. Très souvent, les produits de messagerie incorporent des traitements de texte simplifiés pour composer les messages. La forme la plus achevée de messagerie consiste à gérer les documents composites incluant de la voix, du texte et de l'image, et d'appeler dynamiquement, lors de la réception, les outils de bureautique capables de traiter les contenus du message. L'absence de normalisation du format des données traitées par les outils de bureautique impose de disposer d'une large variété d'outils de visualisation de ces différents formats. La richesse d'un système de messagerie pour le client peut se juger au nombre des outils de visualisation qu'il contient.

Remarque

Moins connue, car non accessible directement par un utilisateur sur son poste de travail, la messagerie entre applications est une composante des systèmes de messagerie. Elle permet à une application de construire des messages qui seront repris ultérieurement par une autre pour être traités. Enfin, il peut y avoir des échanges entre applications et personnes. C'est le cas d'applications qui émettent automatiquement des alarmes à destination d'opérateurs qui devront intervenir au reçu de ces messages.

Remarque

La référence MIME (*MultiPurpose Mail Extension*, RFC 822) définit les règles de codage pour les textes (ASCII à 7 ou 8 bits) et pour les autres documents (jpeg, mpeg…). Historiquement, c'est lors de la définition de la messagerie électronique normalisée (normes *X.400* de l'ISO) qu'on s'est préoccupé pour la première fois du codage des informations transportées. En effet, il eût été dommage d'avoir conçu toute une architecture de protocoles avec fiabilisation des échanges si les données transportées n'avaient plus aucune signification pour le destinataire. Il n'y a pas de norme ou de standard de représentation interne des informations dans les processeurs. Les constructeurs font leur propre choix (taille des mots, place du bit de poids fort…). Si les représentations internes diffèrent, il est impossible de transférer bit à bit les données. L'idée proposée dans *X.400* et reprise plus tard dans les autres messageries est de décrire les règles de codage de l'information et d'adjoindre la référence à ces règles aux données elles-mêmes.



Dans cette section, nous décrivons les trois grands protocoles utilisés dans la messagerie électronique sur Internet : SMTP (*Simple Mail Transfer Protocol*) pour les échanges entre serveurs de messagerie, POP (*Post Office Protocol*) et IMAP (*Internet Message Access Protocol*) pour la communication entre l'utilisateur et le serveur de messagerie qui lui est associé.

4.1 Système de messagerie SMTP (*Simple Mail Transfer Protocol*)

Les messageries SMTP s'appuient sur les fonctions de transport et de réseau de TCP/IP. On les utilise beaucoup dans le cadre d'Internet et elles tendent à se généraliser aussi dans les intranets. La première version opérationnelle de SMTP remonte à 1982 (RFC 821). Les protocoles d'échange ont pour objectif la mise à disposition des messages dans les boîtes aux lettres des destinataires. Trois modes de fonctionnement des boîtes aux lettres existent : en ligne, hors ligne et déconnecté.

- *Mode en ligne.* Une connexion est établie entre l'application cliente et le serveur qui contient la boîte aux lettres.
- *Mode hors ligne.* L'application cliente télécharge les messages vers sa boîte aux lettres, et peut ensuite les manipuler localement tandis que les messages sont effacés du serveur.
- *Mode déconnecté.* L'application cliente rapatrie tout ou partie de ses messages sans les supprimer du serveur, pour les lire localement et conserver cohérente sa boîte aux lettres à sa prochaine connexion.

La norme SMTP prévoyait à l'origine un mode de fonctionnement en ligne ; une première adjonction connue sous le nom de POP a défini un mode hors ligne. Enfin, une extension plus récente IMAP permet les trois modes de fonctionnement. Nous décrivons brièvement les caractéristiques des trois protocoles SMTP, POP3 et IMAP4.

Protocole SMTP

Les échanges SMTP s'appuient sur un réseau TCP/IP et les serveurs de messagerie SMTP utilisent le port 25. Ce protocole prévoit une ouverture du dialogue entre les systèmes SMTP client et serveur, avec identification de l'émetteur et des destinataires du message.

Ces services de base sont mis en œuvre à travers un jeu de commandes simples :

- La *commande HELO* permet à l'émetteur de s'identifier et d'ouvrir le dialogue.
- Les *commandes MAIL, RCPT et DATA* lui permettent de donner l'adresse de sa boîte aux lettres au système destinataire, de vérifier l'existence des boîtes aux lettres des destinataires et d'envoyer les données du message.
- La *commande QUIT* termine le dialogue.

Protocole POP3 (*Post Office Protocol*)

Le protocole POP3 (version 3 de POP, RFC 1939) s'appuie sur un réseau TCP/IP ; un serveur POP3 utilise le port 110. POP3 sert à l'utilisateur client pour rapatrier des messages. La relation entre le client et le serveur prend alors trois états : autorisation, transaction et mise à jour.

- *Autorisation*. Le client ouvre une connexion TCP et s'authentifie.
- *Transaction*. Le client manipule les messages de la boîte aux lettres.
- *Mise à jour*. La connexion TCP est close si le client a demandé de quitter (commande QUIT) ; les messages rapatriés par le client sont alors supprimés de la boîte aux lettres du serveur.

Protocole IMAP4 (*Internet Message Access Protocol*)

Plus évolué que le protocole POP3, IMAP4 (version 4 d'IMAP, RFC 1733 et 2060) standardise les fonctions de manipulations de la boîte aux lettres en réception. Il s'appuie sur un réseau TCP/IP ; un serveur IMAP utilise le port 143.

Le protocole IMAP est un véritable protocole client/serveur qui fournit au client un grand nombre de fonctions de réception de courrier ou d'administration des boîtes aux lettres. Il se rapproche ainsi des fonctions qu'on trouvait dans l'environnement *X.400*, application de messagerie normalisée de l'ISO.

La norme prévoit qu'une session IMAP passe par une succession d'états qui délimitent les jeux de commandes autorisées : authentification, authentifié, sélection et fermeture en cours.

- *Authentification.* Nécessite l'ouverture de la connexion TCP et permet une commande d'authentification.
- *Authentifié.* L'étape précédente a été franchie, le protocole passe dans l'état authentifié : le client sélectionne ou administre une boîte aux lettres.
- *Sélection.* Correspond à la manipulation du contenu de la boîte aux lettres que l'utilisateur client a choisie.
- *Fermeture en cours.* État atteint dès que le client demande à quitter l'application.

Enfin, les messages dans la boîte peuvent être étiquetés : message lu au moins une fois par l'utilisateur ; message auquel l'utilisateur a répondu ; message marqué pour attirer l'attention ; message supprimé logiquement ; message à l'état de brouillon ; message nouveau (qui n'était pas dans la boîte aux lettres lors d'une connexion antérieure). Ces différentes étiquettes du message sont bien utiles pour le client et font le succès des divers produits de messagerie pour le client.

Remarque

On appelle *serveur Webmail* un serveur qui permet aux utilisateurs d'accéder à leurs boîtes aux lettres à travers n'importe quel navigateur. On encapsule alors le trafic de consultation et de manipulation des messages de la boîte dans le protocole HTTP (ou HTTPS si la connexion est sécurisée ; voir les compléments pédagogiques, sur le site www.pearsoneducation.fr).

4.2 Les listes de diffusion

Il est possible de créer des groupes d'usagers avec une seule adresse de messagerie. Un message envoyé par un usager du groupe est reçu par tous les membres du groupe. Un usager peut s'inscrire (et se désinscrire) dans une liste dite *ouverte* ou solliciter son inscription auprès de l'administrateur d'une liste *contrôlée*. Dans les cas où la sécurité est le critère primordial, une liste peut être *fermée*, c'est-à-dire créée et gérée par son propriétaire exclusivement.

Selon les cas, la liste peut être *modérée*, s'il y a un utilisateur particulier, le *modérateur*. Celui-ci filtre les messages avant leur diffusion pour veiller au respect de la politesse, des bons usages… La gestion matérielle de la liste est confiée au serveur de messagerie du domaine concerné. Un utilisateur peut participer à de nombreuses listes de diffusion pour se tenir au courant de l'actualité des divers groupes. Il a alors intérêt à structurer sa boîte aux lettres en sous-dossiers et à y mettre fréquemment de l'ordre, sinon l'espace utilisé par la boîte aux lettres devient rapidement gigantesque.

5 Navigation sur le Web

Le Web est l'application qui a fait l'immense succès d'Internet. En effet, Internet était initialement consacré à des échanges de données entre scientifiques et possédait deux applications : le transfert de fichiers, avec FTP, et la messagerie, avec SMTP et des outils rudimentaires de recherche d'information comme Gopher.

Tim Berners-Lee a inventé le Web en 1989 quand il a proposé un moyen simple de mettre à disposition des fichiers sur le réseau informatique de son entreprise (le CERN, Centre d'études et de recherche nucléaires[4]). Ses travaux débouchèrent sur le standard de présentation des pages appelé HTML (*HyperText Markup Language*) et le protocole de transfert des pages HTTP (*HyperText Transfer Protocol*). L'apparition en 1993 de Mosaïc, le premier navigateur, fut l'élément déclenchant.

Le Web sert à afficher des pages de texte mis en forme à partir de commandes simples et provenant de n'importe quelle machine du réseau. Ce standard permet l'hypertexte (concept né avec le logiciel Hypercard d'Apple), c'est-à-dire l'accès, à partir d'une page, à d'autres pages, au gré du client, indépendamment de l'organisation des pages elles-mêmes.

Chaque page, appelée page Web, est un fichier repéré par une adresse spécifique appelée URL (*Uniform Resource Locator*). Le concepteur d'une page place dans un fichier le texte avec sa mise en forme. Il peut associer à tout mot du texte un pointeur d'adresse URL quelconque, également stocké dans le fichier. Avec cette association, le mot devient en quelque sorte un mot clé. La consultation d'une page Web consiste à transférer le fichier associé (cela est possible à partir de la connaissance de l'adresse URL) contenant le texte et des pointeurs d'adresse URL. Les mots clés sont alors affichés de façon particulière à l'écran (en bleu et souligné au début du Web, puis, maintenant, grâce à un simple changement de forme du curseur de la souris quand il atteint le mot en question, par exemple). Il suffit pour l'usager de cliquer sur un mot clé pour provoquer le rapatriement de la page associée (l'adresse URL étant connue, l'opération ne pose aucun problème).

Un pointeur d'adresse URL peut indiquer non seulement un fichier texte mais aussi des fichiers combinés avec des processus standard de codage d'image (par exemple JPEG) ou de son (loi μ de codage du son à 64 kbit/s). Le Web permet donc le multimédia. La référence MIME (comme pour la messagerie électronique) indiquera dans l'en-tête de la page Web si celle-ci contient du texte, de l'audio, de la vidéo ou un autre format.

Exemple

> L'URL *http://www.linux.org/news/2005/index.htm* représente le lien vers un fichier d'un serveur Web. *http* désigne le nom du protocole de transfert des données ; *www.linux.org* est le nom symbolique de la machine contactée, *news* est un répertoire de cette machine, *2005* est un sous-répertoire du répertoire news. Enfin, *index.htm* est le nom du fichier écrit en langage html. Autrement dit, le fichier recherché est situé dans le sous-répertoire 2005 du répertoire news de la machine www.linux.org. Une URL peut contenir des informations complémentaires comme des mots de passe ou des numéros de port, lorsque les serveurs utilisent des techniques d'identification des clients ou des numéros de ports particuliers.

Un pointeur d'adresse URL peut désigner des pages Web stockées sur d'autres sites. Comme Internet offre un service sans connexion, le passage d'une page stockée sur un ordinateur d'une université française à une page d'un ordinateur d'une entreprise en Australie peut se faire très rapidement. Il est donc possible de naviguer ou de *surfer* sur le réseau et de voyager virtuellement à travers le monde.

Les utilisateurs ont rapidement détourné l'application Web de sa mission initiale, quand la publicité, les échanges commerciaux et financiers sont apparus… et les problèmes de sécurité. Nombre d'entreprises poussent même leurs clients à consulter des informations sur Internet, à y réaliser des achats et des paiements électroniques. Le Web est devenu le premier support d'information aujourd'hui et les réseaux des entreprises en ont copié la convivialité : un intranet n'est autre qu'un réseau d'entreprise qui utilise les technologies du Web, de même un extranet est un intranet ouvert aux clients et fournisseurs de l'entreprise.

4. Le plus grand laboratoire de physique des particules au monde.

5.1 Architecture du Web

Le Web repose sur une architecture client/serveur. Le serveur met des informations à disposition. Le client, avec son navigateur, se connecte pour demander à recevoir ces informations qui s'affichent sur son écran. Le protocole de transport sous-jacent est TCP. Dans la version initiale de HTTP, la connexion était rompue après chaque échange d'information : une page Web qui contient trois images est transférée en quatre connexions TCP ce qui représente un grand nombre de requêtes/réponses pour le serveur. La version HTTP1.1 regroupe le transfert de tous les éléments d'une page dans une seule connexion. Le serveur Web reçoit une demande d'ouverture de connexion sur le port 80 (par défaut), puis une fois la connexion établie, une requête GET qui demande un document particulier. Le serveur recherche dans son environnement de stockage le document en question et crée le flux de données correspondant au contenu du document décrit en HTML et, enfin, la connexion avec le client est rompue. La figure 9.4 illustre les échanges entre client et serveur.

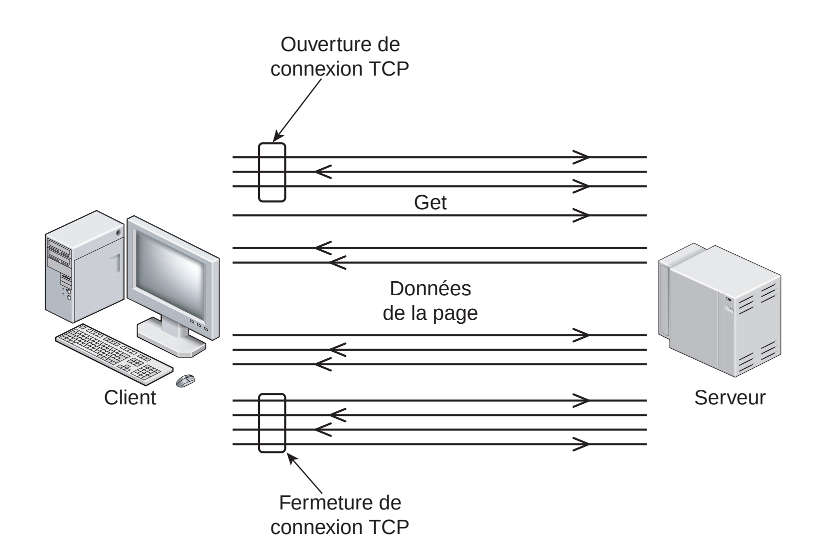


Figure 9.4
Échanges HTTP.

Le premier produit serveur sur le marché est indéniablement Apache, un produit sous licence GNU qu'on peut télécharger gratuitement. Pour le client, le navigateur regroupe les fonctions de base d'une application avec son paramétrage, une barre d'outils pour naviguer dans les pages déjà reçues (revenir à une page déjà vue, imprimer une page…) et une barre d'adresse où le client entre l'URL du document recherché. Le navigateur interprète les données reçues et les affiche selon ses capacités graphiques. De nombreux navigateurs sont proposés sur le marché pour les clients.

5.2 Autour du Web

Pages dynamiques et animation des pages Web

Certaines pages Web sont créées spécialement en réponse à la requête d'un utilisateur (ou d'un client). Elles possèdent alors une forme et un contenu variables, adaptés à ses besoins. On parle de pages *dynamiques*. Elles utilisent le plus souvent le langage JavaScript (qui n'a rien à voir avec Java). JavaScript repose sur une programmation à événements et nécessite donc que le navigateur du client supporte l'exécution de code associée à des événements. Une image qui change au moment où la souris passe dessus est un exemple simple d'exécution de code associé à un événement.



Les navigateurs sont maintenant présents sur toutes les machines des clients. Néanmoins, ils ne sont pas tous capables d'interpréter les mêmes formats de données. Certains fournisseurs de logiciels ont proposé l'idée de *plug-in*, composants logiciels supplémentaires, téléchargeables et exécutables par le navigateur pour en améliorer les capacités (par exemple, un plug-in Acrobate Reader pour afficher correctement des fichiers .pdf, un plug-in RealPlayer pour visualiser et écouter des flux vidéo dans une page Web…). Ces ajouts sont malheureusement souvent des solutions « propriétaires ». Microsoft, qui a inclus son navigateur phare dans son système d'exploitation Windows, a de même proposé des composants logiciels baptisés *ActiveX* téléchargeables, automatiquement installés dans le système d'exploitation et référencés dans la base de registres Windows.

La solution la plus fréquente pour animer les pages Web consiste à utiliser de petites applications écrites en langage Java (les *applets* Java), qui en exploitent les capacités d'animation et de rendu graphique. Elles sont contenues dans la page Web. On les télécharge dynamiquement en même temps que la page qui les référence. Le navigateur contrôle ensuite leur exécution. Le téléchargement se fait en mémoire vive à chaque visite de la page. Rien n'est installé sur le poste client. Cela procure de moindres performances par rapport aux ActiveX mais protège mieux le client de toute installation indésirable sur son système.

L'ensemble des technologies de programmation des serveurs Web, depuis les premiers CGI (*Common Gateway Interface*), qui permettaient la création de formulaire où l'utilisateur peut remplir les différents champs et déclencher des actions en fonction des informations saisies, jusqu'aux ASP (*Active Server Pages* de Microsoft) ou JSP (*Java Server Pages*), déborde du cadre de cet ouvrage. Citons simplement, en marge des produits commerciaux, le succès des solutions construites autour de produits gratuits d'excellente qualité PHP et MySQL qui s'exécutent côté serveur, interfacent les pages Web avec des bases de données et construisent dynamiquement les pages affichées à l'utilisateur.

Cookies et émulation de connexion

Pour les sites commerciaux, il est normal que tous les produits proposés ne soient pas sur la même page Web. Le fonctionnement non connecté d'HTTP qui referme la connexion TCP après le chargement de chaque page obligerait le client à fournir son identité et ses références bancaires chaque fois qu'il change de page : le serveur « oublie » le client une fois qu'il a traité sa requête ! On utilise alors des techniques d'*émulation de connexion*, qui consistent à associer un client à un identifiant unique (improprement appelé *clé de session*). Le navigateur du client transmet cet identifiant à chaque requête vers le serveur. L'une des solutions consiste à utiliser des *cookies*. Ce sont des petits fichiers de texte engendrés par le serveur dès qu'un client visite le site. Ils sont stockés sur les machines des utilisateurs, à leur insu. Le serveur les exploite lors du changement de pages et lors des connexions suivantes des clients. Certains sites marchands vont jusqu'à conserver un profil de chaque client en repérant ses habitudes de navigation et d'achats. Les cookies permettent donc de gérer un contexte vis-à-vis du client, sorte de session virtuelle, dans laquelle le client est heureux de retrouver les pages du serveur avec les paramètres de présentation qu'il peut avoir choisis lui-même.

L'avantage des cookies par rapport aux autres solutions (comme les URL longues ou les trames cachées) est qu'ils durent au-delà de la visite du site par le client : l'utilisateur les stocke lui-même sur son disque. Ils disposent d'une date de péremption. L'utilisateur peut évidemment effacer les cookies, d'autant plus qu'une utilisation parfois abusive par certains sites commerciaux leur a donné la réputation d'instruments d'intrusion dans la vie privée du client. De plus, l'ensemble des cookies finit à la longue par encombrer le disque et peut ralentir le fonctionnement du navigateur.

Si un utilisateur surfe sur Internet en utilisant plusieurs ordinateurs différents, il doit s'identifier sur chaque ordinateur. Il a intérêt à ne pas accepter les cookies permanents et à effacer tous les fichiers temporaires car un autre utilisateur sur le même ordinateur pourrait ensuite se faire passer pour lui.

Sites particuliers

Les *moteurs de recherche* (Google, Yahoo!, Voila, AltaVista…) sont des serveurs spécialisés dans la recherche d'informations à partir de mots clés. Ils contiennent des banques de données textuelles alimentées en permanence par des programmes automatiques d'indexation. Ces programmes explorent tous les sites et regroupent par thèmes les informations recueillies. Ces outils de recherche ont une puissance impressionnante : ils peuvent trouver des milliers de pages avec les mots clés d'une recherche en moins d'une seconde. Le problème pour l'utilisateur est alors de bien cibler sa recherche, en précisant ses mots clés. Des *métamoteurs* (Copernic…) ont été proposés, ils offrent l'intérêt de rechercher directement sur plusieurs moteurs de recherche, de comparer l'ensemble des résultats et d'éliminer les références en double.

Un *blogue* (en anglais *blog*, contraction de *weblog*), est un site Web personnel, évolutif et souvent non conformiste, présentant des réflexions de toutes sortes, généralement sous forme de courts messages. L'auteur met à jour son blogue aussi souvent qu'il le veut, ses lecteurs peuvent y apporter leurs commentaires.

À la différence d'un blogue, qui exprime la pensée d'un individu, le *wiki* est un site Web collaboratif, matérialisant les idées d'un groupe qui partage des intérêts communs. Le nom *wiki* provient de l'adjectif hawaiien *wikiwiki*, qui signifie rapide. Ward Cunningham a inventé ce système en 1995, son premier site utilisant ce principe était WikiWikiWeb. Afin qu'un site Web puisse offrir ces services, un moteur wiki personnalisable doit être installé. Pour modifier les informations du site, il suffit en général de cliquer sur un lien « modifier », chacun apporte ainsi sa contribution. Pour l'édition des pages, une syntaxe très simplifiée héritée de HTML traite la mise en page et l'activation des liens. Le site est amené à grossir sans qu'un webmestre le décide, mais simplement parce que les utilisateurs l'ont souhaité. Il faut toutefois qu'un utilisateur procède de temps à autre à un nettoyage, pour éviter le désordre : suppression de pages obsolètes, de contenus redondants, etc. Les wikis ont un très grand succès auprès du public et des communautés comme les étudiants ou les employés d'une entreprise en créent très souvent. À titre d'exemple, l'encyclopédie Wikipedia, qui existe dans une trentaine de langues, compte plus de 350 000 pages, correspondant à plus de 180 000 articles.

Résumé

Nous avons présenté au cours de ce chapitre quelques applicatifs utilisant TCP/IP parmi les plus répandus. Le service de configuration dynamique des machines et le service de noms de domaine sont des applications internes, utiles au bon fonctionnement des réseaux. Ils fournissent aux machines toutes les informations indispensables pour pouvoir communiquer dans leur réseau et sur Internet. Le premier fournit l'adresse IP et les informations minimales de configuration (masque de sous-réseau, adresse du routeur pour sortir du réseau, adresse du serveur de noms…). Le second est indispensable pour exploiter toute la richesse des informations sur Internet. Il permet la conversion automatique des noms de machines en adresses IP, nécessaires pour toutes les communications à travers le réseau. Les applications concernant directement les utilisateurs et leurs besoins de communication à travers les réseaux sont la messagerie électronique, le transfert de fichiers et enfin la navigation sur le Web. Nous avons présenté succinctement les protocoles associés à chacune de ces applications ainsi que leurs caractéristiques principales, sans aborder les problèmes de sécurité ou d'intégration des postes mobiles qui seront vus au chapitre suivant.



Problèmes et exercices

Exercice 1 Utilisation de DHCP

Énoncé

Sur un réseau Ethernet, une machine client se connecte pour la première fois et diffuse un message *DHCPDiscover* pour trouver un serveur DHCP.

a Quels sont les numéros de port utilisés dans le datagramme UDP transportant ce message ?

b Quelles sont les adresses IP émetteur et destinataire dans le datagramme IP ?

c Quelles sont les adresses MAC utilisées dans la trame Ethernet qui encapsule le datagramme IP ?

Solution

a Les numéros de port sont 68 pour le client et 67 pour le serveur.

b La machine n'ayant pas d'adresse IP, elle laisse à « plein 0 » le champ adresse IP source et comme elle ne connaît pas non plus l'adresse du destinataire, elle utilise l'adresse de diffusion « plein 1 ». Nous avons donc :

– adresse IP source = `0.0.0.0`, port source 68 ;

– adresse IP destination = `255.255.255.255`, port destination 67.

c Le datagramme IP est encapsulé dans une trame Ethernet dont l'adresse MAC de destination est `FF:FF:FF:FF:FF:FF` puisqu'il s'agit d'une diffusion. L'adresse MAC source est le numéro de série de la carte réseau de la machine en question.

Exercice 2 Redondance de serveurs DHCP

Énoncé

On considère un réseau constitué de trois sous-réseaux interconnectés par le même routeur. Un serveur DHCP est installé dans le premier sous-réseau et un second dans le deuxième sous-réseau.

a Une machine du troisième sous-réseau peut-elle obtenir une adresse IP en utilisant DHCP ? Si oui comment ? Sinon, pourquoi ?

b Pourquoi avoir installé plusieurs serveurs DHCP ?

Solution

a Une machine du troisième sous-réseau peut obtenir une adresse IP en utilisant DHCP à condition que le routeur implémente un agent relais qui transfère le message en diffusion (`255.255.255.255`) au-delà du routeur. Les deux serveurs DHCP reçoivent alors la requête *DHCPDiscover*.

b Deux serveurs DHCP ont été installés pour une meilleure fiabilité : si l'un est en panne, l'autre est opérationnel.

Exercice 3 Rôle d'un serveur DNS et trafic interne

Énoncé

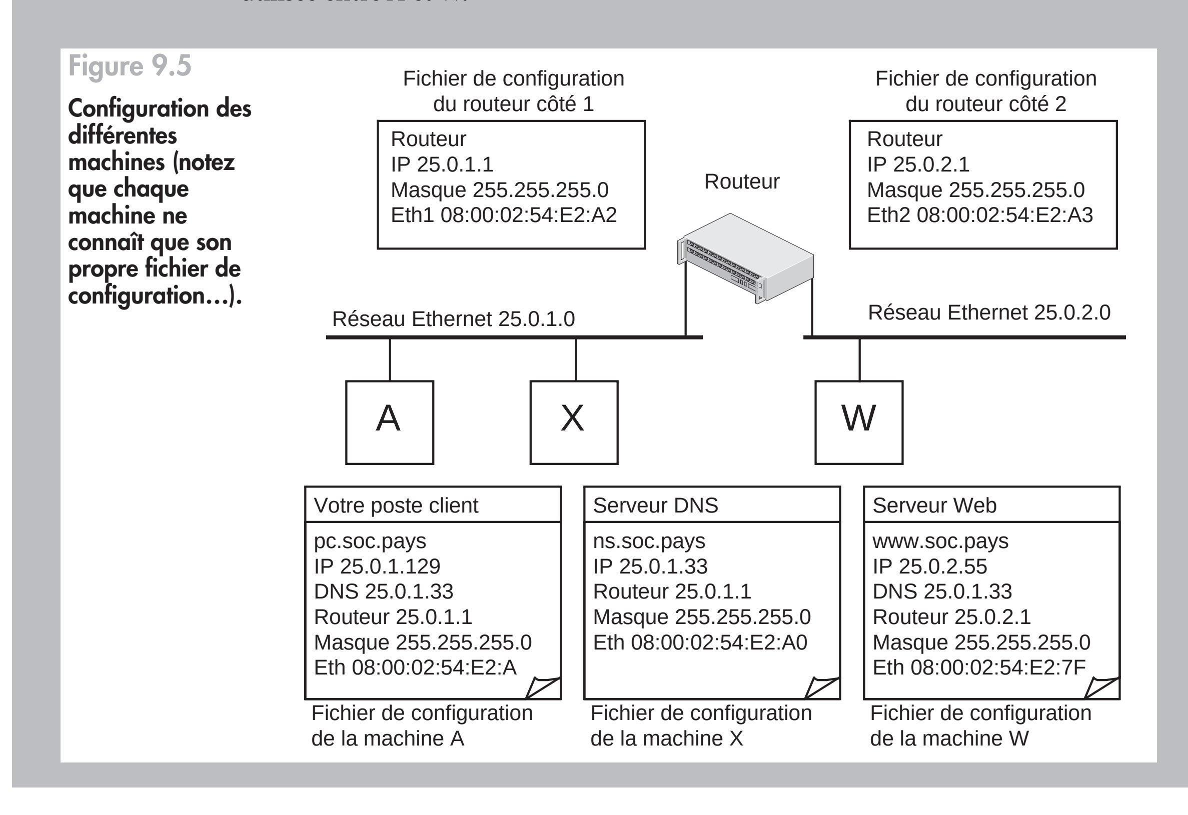
On considère un réseau composé de trois ordinateurs (un PC, un serveur DNS et un serveur Web), un routeur et deux réseaux locaux de type Ethernet comme le montre la figure 9.5. L'architecture de protocoles utilisée est TCP/IP.

Donnez les différentes trames échangées sur les deux réseaux Ethernet lorsque la machine *A* (`pc.soc.pays`) cherche à établir une session www sur le serveur *W* (`www.soc.pays`). On supposera que les trois machines et le routeur sont correctement configurés. Ils viennent d'être installés (tous les caches sont vides) et sont allumés lorsque le client commence sa requête. On supposera également que la machine *X* (`ns.soc.pays`) se trouvant sur le même réseau que le client dispose de l'information permettant de résoudre directement le nom `www.soc.pays` en une adresse IP.

La réponse se présentera dans l'ordre chronologique. On indiquera pour chaque trame le réseau sur lequel elle a été émise (1 pour le réseau `25.0.1.0` et 2 pour le réseau `25.0.2.0`), les adresses physiques de la source et de la destination de la trame, les adresses IP de la source et de la destination (si nécessaire) et un bref commentaire sur le contenu ou la fonction de cette trame ou de son contenu. La dernière trame à indiquer est celle contenant l'arrivée de la confirmation d'établissement de la connexion TCP utilisée entre *A* et *W*.

Figure 9.5

Configuration des différentes machines (notez que chaque machine ne connaît que son propre fichier de configuration…).



Solution

La machine *A* doit en premier lieu obtenir du serveur DNS la conversion du nom symbolique www.soc.pays en adresse IP. Son fichier de configuration lui fournit l'adresse IP du serveur DNS à interroger, il faut donc tout d'abord émettre une requête ARP pour obtenir son adresse MAC.



Trame 1 sur réseau 1 = trame Ethernet diffusée par A (@MAC *A* vers @MAC FF:FF:FF:FF:FF:FF). Cette trame contient une requête ARP (champ protocole = 0805) pour connaître l'adresse MAC du serveur DNS que *A* connaît seulement par son adresse IP. Le serveur DNS qui a reçu cette trame et reconnu son adresse IP répond.

Trame 2 sur réseau 1 = trame Ethernet (@MAC *X* vers @MAC *A*) contenant la réponse ARP fournissant l'adresse MAC du serveur DNS. *A* inscrit dans sa table ARP la correspondance @IP 25.0.1.33 = @MAC 08:00:02:54:E2:A0, et maintenant que *A* connaît l'adresse MAC de *X*, elle peut lui envoyer une trame Ethernet.

Trame 3 sur réseau 1 = trame Ethernet (@MAC *A* vers @MAC *X*). Cette trame contient un datagramme IP (@IP *A* vers @IP *X*). Le datagramme contient un message UDP (port distant 53) contenant la requête au DNS (« je recherche l'adresse IP de www.soc.pays »).

Trame 4 sur réseau 1 = trame Ethernet (@MAC *X* vers @MAC *A*). Cette trame contient un datagramme IP (@IP *X* vers @IP *A*). Le datagramme contient un message UDP (port local 53) portant la réponse du DNS (www.soc.pays = @IP 25.0.2.55). *A* connaît maintenant l'adresse IP de son correspondant, en l'occurrence le serveur Web. En utilisant le masque de sous-réseau qui est présent dans son fichier de configuration, *A* constate que le serveur Web n'est pas dans le même (sous-)réseau que lui. Il faudra donc passer par le routeur pour sortir du (sous-)réseau. Or le routeur n'est connu (fichier de configuration de *A*) que par son adresse IP que nous noterons @IP *R1* : il faut procéder à une nouvelle requête ARP pour obtenir son adresse MAC.

Remarque

On pourrait se demander pourquoi le fichier de configuration contient l'adresse IP du routeur alors que c'est l'adresse MAC du routeur qui sert, celui-ci étant sollicité localement pour chaque message qui doit sortir du réseau. Fournir l'adresse IP est une solution souple qui permet de changer l'équipement matériel du routeur sans avoir à reconfigurer toutes les machines… Nous avons vu au chapitre 5 qu'on utilise aujourd'hui la notion d'adresse IP virtuelle de routeur qui apporte une souplesse supplémentaire.

Trame 5 sur réseau 1 = trame Ethernet diffusée par *A* (@MAC *A* vers @MAC FF:FF:FF:FF:FF:FF). Cette trame contient une requête ARP pour connaître l'adresse MAC du routeur connu par @IP *R1*.

Trame 6 sur réseau 1 = trame Ethernet (@MAC *R1* vers @MAC *A*). Cette trame contient la réponse ARP fournissant l'adresse MAC du routeur (côté sous-réseau 1). *A* inscrit dans sa table ARP la correspondance @IP 25.0.1.1 = @MAC 08:00:02:54:E2:A2 et maintenant que *A* connaît l'adresse MAC du routeur (notée @MAC *R1*), elle peut lui envoyer une trame Ethernet.

Trame 7 sur réseau 1 = trame Ethernet (@MAC *A* vers @MAC *R1*). Cette trame contient un datagramme IP (@IP *A* vers @IP *W*) qui contient un segment TCP de demande d'ouverture de connexion (drapeau SYN) pour HTTP (port 80).

Le routeur a reçu la trame 7 puisqu'elle lui était adressée. Il en a décapsulé le datagramme IP et, après consultation de sa table de routage, a constaté que le réseau de *W* était joignable directement sur sa deuxième interface. Nous faisons ici l'hypothèse que l'adresse MAC de *W* ne figure pas dans la table ARP du routeur.

Trame 8 sur réseau 2 = trame Ethernet diffusée par le routeur (@MAC *R2* vers @MAC FF:FF:FF:FF:FF:FF). Cette trame contient une requête ARP pour connaître l'adresse MAC de *W* dont le routeur ne connaît que @IP *W*.

Trame 9 sur réseau 2 = trame Ethernet (@MAC *W* vers @MAC *R2*). Cette trame contient la réponse ARP fournissant l'adresse MAC de *W*. Le routeur inscrit dans sa table ARP la correspondance @IP `25.0.2.55` = @MAC `08:00:02:54:E2:7F` et maintenant qu'il connaît l'adresse MAC de *W*, il peut lui envoyer une trame Ethernet.

Trame 10 sur réseau 2 = trame Ethernet (@MAC *R2* vers @MAC *W*). Cette trame contient le datagramme IP (@IP *A* vers @IP *W*), qui renferme le segment TCP de demande d'ouverture de connexion pour HTTP (port 80). Ce datagramme est celui qui était dans la trame 7, la seule différence est le champ TTL que le routeur a réduit de 1 et donc le bloc de contrôle d'erreur sur l'en-tête qui a été de ce fait recalculé.

Trame 11 sur réseau 2 = trame Ethernet diffusée par *W* (@MAC *W* vers @MAC `FF:FF:FF:FF:FF:FF`) contenant une requête ARP pour connaître l'adresse MAC du routeur (*A* est dans un autre sous-réseau).

Trame 12 sur réseau 2 = trame Ethernet (@MAC *R2* vers @MAC *W*). Cette trame contient la réponse ARP fournissant l'adresse MAC du routeur (côté sous-réseau 2).

Trame 13 sur réseau 2 = trame Ethernet (@MAC *W* vers@MAC *R2*). Cette trame contient un datagramme IP (@IP *W* vers @IP *A*), qui possède un segment TCP de réponse positive à la demande d'ouverture de connexion (drapeaux SYN et ACK) pour http.

Trame 14 sur réseau 1 = trame Ethernet diffusée par *R1* (@MAC *R1* vers @MAC `FF:FF:FF:FF:FF:FF`). Cette trame contient la requête ARP pour connaître l'adresse MAC de *A*.

Trame 15 sur réseau 1 = trame Ethernet (@MAC *A* vers@MAC *R1*). Cette trame contient la réponse ARP fournissant l'adresse MAC de *A*.

Trame 16 sur réseau 1 = trame Ethernet (@MAC *R1* vers @MAC *A*). Cette trame contient un datagramme IP (@IP *W* vers @IP *A*), qui renferme le segment TCP de réponse positive à la demande d'ouverture de connexion pour HTTP. Ce datagramme est celui qui a été transporté dans le réseau 2 encapsulé dans la trame 13, aux champs TTL et bloc de contrôle d'erreur près.

Trame 17 sur réseau 1 = trame Ethernet (@MAC *A* vers @MAC *R1*). Cette trame contient un datagramme IP (@IP *A* vers @IP *W*), lequel recèle un segment TCP de confirmation d'ouverture de connexion (drapeau ACK) pour HTTP (port 80).

Trame 18 sur réseau 2 = trame Ethernet (@MAC *R2* vers @MAC *W*). Cette trame contient le datagramme IP (@IP *A* vers @IP *W*), qui contient le segment TCP de confirmation d'ouverture de connexion (drapeau ACK) pour HTTP (port 80).

Remarque

Cet exercice est volontairement détaillé. L'objectif était de montrer l'ensemble du trafic généré par la recherche des adresses MAC et par l'utilisation du serveur DNS. D'autre part, il illustre la notion d'encapsulation en insistant sur le fait que les requêtes HTTP ou DNS sont transmises dans des messages de la couche 4 (TCP ou UDP), lesquels sont véhiculés par les datagrammes IP. Ci-après, vous trouverez une version simplifiée dans laquelle on suppose que tous les caches ARP contiennent les correspondances nécessaires : on enlève tout le trafic ARP.

Il ne reste que deux phases d'échange, l'interrogation de l'annuaire et l'ouverture de la connexion TCP avec le serveur Web.



a. Interrogation de l'annuaire :

- Trame 3 sur réseau 1 = trame Ethernet (de *A* vers *X*) contenant un datagramme IP (de *A* vers *X*) contenant un message UDP contenant la requête au DNS (recherche adresse IP de *W* ?).
- Trame 4 sur réseau 1 = trame Ethernet (de *X* vers *A*) contenant un datagramme IP (de *X* vers *A*) contenant un message UDP contenant la réponse du DNS (avec adresse IP de *W*).

b. Ouverture de connexion TCP en trois temps avec le serveur Web :

- Trame 7 sur réseau 1 = trame Ethernet (de *A* vers *R1*) contenant un datagramme IP (de *A* vers *W*) contenant un message TCP de demande d'ouverture de connexion (drapeau SYN) pour HTTP (port 80). On traverse le routeur.
- Trame 10 sur réseau 2 = trame Ethernet (de *R2* vers *W*) contenant le datagramme IP (de *A* vers *W*) contenant un message TCP de demande d'ouverture de connexion (drapeau SYN) pour HTTP (port 80).
- Trame 13 sur réseau 2 = trame Ethernet (de *W* vers *R2*) contenant un datagramme IP (de *W* vers *A*) contenant un message TCP de réponse positive à la demande d'ouverture de connexion (drapeaux SYN et ACK) pour HTTP. On traverse à nouveau le routeur.
- Trame 16 sur réseau 1 = trame Ethernet (de *R1* vers *A*) contenant un datagramme IP (de *W* vers *A*) contenant un message TCP de réponse positive à la demande d'ouverture de connexion (drapeaux SYN et ACK) pour HTTP.
- Trame 17 sur réseau 1 = trame Ethernet (de *A* vers *R1*) contenant un datagramme IP (de *A* à *W*) contenant un segment TCP de confirmation d'ouverture de connexion (drapeau ACK) pour HTTP. On traverse encore le routeur.
- Trame 18 sur réseau 2 = trame Ethernet (de *R2* vers *W*) contenant le datagramme IP (de *A* à *W*) contenant le segment TCP de confirmation d'ouverture de connexion (drapeau ACK) pour HTTP (port 80).

Enfin, si on ne donne que la vision applicative, il y a deux échanges.

a. Interrogation de l'annuaire : requête de *A* au DNS (recherche adresse IP de *W*) et réponse du DNS (avec adresse IP de *W*).

b. Ouverture de connexion TCP en trois temps avec le serveur Web :

- demande d'ouverture de connexion de *A* (drapeau SYN) pour *W* (port 80) ;
- réponse positive à la demande d'ouverture de connexion de *W* (drapeaux SYN et ACK) pour *A* ;
- confirmation d'ouverture de connexion de *A* (drapeau ACK) pour *W*.

Exercice 4 Serveur DHCP et serveur DNS

Énoncé

Cet exercice est le même que le précédent mais on suppose que l'adresse utilisée dans le réseau est `172.25.0.0` avec un masque `255.255.192.0` : le réseau est découpé en deux sous-réseaux et la machine *Y* est un serveur DHCP, situé dans le premier sous-réseau (voir figure 9.6). Le serveur DNS, le serveur Web et le routeur possèdent des adresses fixes, respectivement `172.25.64.33`, `172.25.128.55`, `172.25.64.1` et `172.25.128.1`, les deux dernières étant les deux adresses du routeur. Le serveur DHCP distribue dynamiquement les adresses dans la plage `172.25.64.64` à `172.25.64.191` sur le réseau 1. La première adresse disponible est `172.25.64.75`.

☞

Exercices



Figure 9.6

Configuration étudiée.

Fichier de configuration du routeur côté 1
Routeur
IP 172.25.64.1
Masque 255.255.192.0
Eth1 08:00:02:54:E2:A2

Routeur

Fichier de configuration du routeur côté 2
Routeur
IP 172.25.128.1
Masque 255.255.192.0
Eth2 08:00:02:54:E2:A3

Réseau 172.25.64.0

Réseau 172.25.128.0

A
X
Y
W

dhcp.soc.pays
IP 172.25.64.34
Routeur 172.25.64.1
Masque 255.255.192.0
Eth 08:00:02:54:E2:80
Fichier de configuration de la machine Y

Serveur Web
www.soc.pays
IP 172.25.128.55
DNS 172.25.64.33
Routeur 25.0.2.1
Masque 255.255.192.0
Eth 08:00:02:54:E2:7F
Fichier de configuration de la machine W

ns.soc.pays
IP 172.25.64.33
Routeur 172.25.64.1
Masque 255.255.192.0
Eth 08:00:02:54:E2:10
Fichier de configuration de la machine X

Utiliser DHCP
Eth 08:00:02:54:E2:A1
Fichier de configuration de la machine A

La machine *A* connaît son adresse MAC et sait qu'il faut utiliser DHCP. Décrivez le trafic *supplémentaire* par rapport à l'exercice 3 pour que la machine *A* obtienne son adresse IP avant de démarrer le scénario précédent.

Solution

Par rapport à l'exercice précédent, il faut ajouter le trafic lié à la recherche d'un serveur DHCP et le dialogue avec celui-ci.

Trame I sur réseau 1 = trame Ethernet diffusée par *A* (@MAC *A* vers @MAC `FF:FF:FF:FF:FF:FF`). Cette trame contient un datagramme IP (@IP `0.0.0.0` vers @IP `255.255.255.255`), qui encapsule un message UDP (port 67) contenant le message *DHCPDiscover*.

Trame II sur réseau 1 = trame Ethernet diffusée par *Y* (@MAC *Y* vers @MAC FF:FF:FF:FF:FF:FF). Cette trame contient un datagramme IP (@IP *Y* vers @IP 172.25.64.75), qui encapsule un message ICMP (*Echo Request*). Le serveur DHCP teste si l'adresse IP qu'il veut proposer à *A* est disponible en envoyant un *ping* sur cette adresse. Si l'adresse est disponible, la requête *ping* n'obtient pas de réponse.

Trame III sur réseau 1 = trame Ethernet (@MAC *Y* vers @MAC *A*). Cette trame contient un datagramme IP (@IP *Y* vers @IP 255.255.255.255), qui encapsule un message UDP (port 68) contenant le message *DHCPOffer* avec @IP 172.25.64.75.

Trame IV sur réseau 1 = trame Ethernet diffusée par *A* (@MAC *A* vers @MAC FF:FF:FF:FF:FF:FF). Cette trame contient un datagramme IP (@IP 0.0.0.0 vers @IP 255.255.255.255), qui encapsule un message UDP (port 67) contenant le message *DHCPRequest* : « J'ai choisi le serveur DHCP *Y* avec son offre @IP 172.25.64.75. »

Trame V sur réseau 1 = trame Ethernet (@MAC *Y* vers @MAC *A*). Cette trame contient un datagramme IP (@IP *Y* vers @IP 172.25.64.75), qui encapsule un message UDP (port 68) contenant le message *DHCPAck* : « Voici les autres informations de configuration : masque de sous-réseau, adresse IP routeur par défaut, adresse IP serveur DNS… »

Trame VI sur réseau 1 = trame Ethernet diffusée par *A* (@MAC *A* vers @MAC FF:FF:FF:FF:FF:FF). Cette trame contient une requête ARP : « Je cherche l'adresse MAC de la machine d'adresse IP 172.25.64.75. »

A priori, si tout s'est bien passé (l'adresse IP proposée n'était pas déjà attribuée à une autre machine du réseau), la trame VI n'a pas de réponse puisque l'adresse recherchée est justement celle qui est proposée à la station A.

Les trames I à VI sont échangées avant les trames 1 à 18 de l'exercice précédent.

Exercice 5 Enregistrements sur un serveur DNS

Énoncé

Le paramétrage d'un serveur DNS contient les temporisateurs suivants :

- 21600 ; Refresh
- 3600 ; Retry
- 604800 ; Expire
- 172800 ; Minimum

À quoi correspondent ces différentes durées ?

Solution

Les temporisateurs sont exprimés en secondes : 3600 représente une heure ; 21600 = six heures ; 172800 = deux jours et 604800 = une semaine. *Refresh* est l'intervalle de temps (ici six heures) entre deux vérifications d'un serveur secondaire sur le serveur officiel primaire pour savoir s'il y a eu des modifications et si une mise à jour est nécessaire ; *Retry* est le temps d'attente (ici une heure) d'un serveur secondaire avant de renouveler sa mise à jour si la précédente a échoué ; *Expire* est le temps (ici une semaine) au bout duquel les informations sont jetées si elles n'ont pas pu être mises à jour ; *Minimum* est la durée (ici deux jours) pendant laquelle un serveur DNS peut conserver en cache un enregistrement du fichier de la base de données.

EXERCICE 6 SERVEUR DNS ET CACHE

Énoncé

Soit un serveur DNS qui a pour nom *toutatis.sous_domaine.mon_domaine.mon_pays* qui est serveur primaire de la zone *sous_domaine.mon_domaine.mon_pays* et dont le cache est encore vide.

a Un utilisateur sollicite ce serveur pour obtenir l'adresse IP de *ftp.mon_domaine.mon_pays*. Quel serveur interroge *toutatis* pour répondre ?

b Le même utilisateur cherche maintenant *www.mon_domaine.mon_pays*. Même question !

c Même question maintenant pour *www.autre_domaine.mon_pays* puis pour *ftp.sous_domaine. autre_domaine.mon_pays*.

Solution

a Le serveur DNS interroge un serveur racine pour savoir qui gère *.mon_pays*. Avec la réponse, il interroge le serveur DNS qui gère *mon_pays* pour savoir qui gère *mon_domaine.mon_pays*. Avec la réponse, il interroge enfin le serveur DNS qui gère *mon_domaine* pour récupérer l'adresse IP de *ftp.mon_domaine.mon_pays*. Il met à jour son cache avec les informations récoltées.

b Cette fois-ci, les informations du cache sont utiles. Il suffit d'interroger un serveur DNS qui gère *mon_domaine.mon_pays* pour connaître l'adresse du serveur Web *www.mon_domaine.mon_pays*. Le serveur DNS met à nouveau à jour son cache avec les informations récoltées.

c Le serveur DNS s'adresse au serveur DNS qui gère *mon_pays* (et qu'il connaît déjà) pour savoir qui gère *autre_domaine.mon_pays* puis il interroge ce dernier pour savoir l'adresse du serveur *www.autre_domaine.mon_pays*. Il met encore à jour son cache avec les informations récoltées. Ces informations seront utiles pour la requête suivante. Il suffira d'interroger le serveur DNS connu qui gère *autre_domaine.mon_pays* pour connaître l'adresse du serveur DNS qui gère *sous_domaine.autre_domaine.mon_pays* et demander à ce dernier l'adresse du serveur ftp *ftp.sous_domaine. autre_domaine.mon_pays* . Une dernière fois, il met à jour son cache avec les informations récoltées.

EXERCICE 7 PROTOCOLES DE CONSULTATION DE BOÎTE AUX LETTRES

Énoncé

Établissez un tableau comparatif des deux protocoles POP et IMAP qu'un usager peut utiliser pour consulter sa boîte aux lettres selon les critères suivants : lieu de stockage des messages, espace mémoire nécessaire sur le serveur, possibilité de création de dossiers de courrier, d'interroger sa boîte aux lettres depuis n'importe où, d'utiliser Webmail…

Solution

	POP	**IMAP**
Lieu de stockage des messages	Chez le client dès qu'ils sont transférés	Les messages sont conservés sur le serveur
Espace mémoire sur le serveur	Faible	Grand
Dossiers de courrier	Chez le client	Sur le serveur
Interrogation de la boîte aux lettres de n'importe où	Non	Oui
Possibilité d'utiliser Webmail	Non concerné	Non concerné



Remarque

Un fournisseur d'accès peut avoir intérêt à proposer à ses clients de messagerie l'utilisation d'IMAP même si cela engendre des coûts de stockage importants sur le serveur : il peut proposer de facturer le stockage au-delà de quelques Mo, générant ainsi du chiffre d'affaires.

EXERCICE 8 ANALYSE DE L'EN-TÊTE D'UN COURRIER ÉLECTRONIQUE

Énoncé

Votre outil de messagerie permet d'afficher l'en-tête complet d'un courrier électronique reçu. Commentez l'exemple ci-après en explicitant le chemin par lequel est passé le message.

```
Return-Path: <nom.prenom@free.fr>
Received: from mel-rti17.wanadoo.fr (192.168.156.136) by ms9.wana-
doo.fr; 2 May 2006 08:36:38 +0200
Received: from postfix1-1.free.fr (213.228.0.2) by mel-rti17.wana-
doo.fr
id 3CD0C9B00000BEB5; Thu, 2 May 2006 08:36:33 +0200
Received:     from     postfix1-2.free.fr     (postfix1-2.free.fr
[213.228.0.130])
  by postfix1-1.free.fr (Postfix) with ESMTP
  id E16611027EB; Thu, 2 May 2002 08:36:31 +0200 (CEST)
Received:     from     oemcomputer     (montpellier-1-a7-62-147-81-
254.dial.proxad.net [62.147.81.254])
  by postfix1-2.free.fr (Postfix) with SMTP
  id BA1C1AB595; Thu, 2 May 2006 08:34:32 +0200 (CEST)
Message-ID: <001701c1f1a3$0d1c3440$fe51933e@oemcomputer>
From: "moi" <nom.prenom@free.fr>
To: …
Subject: …
Date: Thu, 2 May 2006 08:27:35 +0200
MIME-Version: 1.0
Content-Type: multipart/related;
  type="multipart/alternative";
  boundary="----=_NextPart_000_0010_01C1F1B3.369BBA60"
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 6.00.2600.0000
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2600.0000
```

Solution

Le message a été émis le 2 mai 2006 à 8 h 27 min 35 s (+ deux heures par rapport à l'heure GMT, ce qui correspond à l'heure d'été en France). Il a été composé avec l'outil de messagerie de Microsoft Outlook Express 6 avec une priorité normale et il a été relayé par plusieurs serveurs de messagerie.

Il faut lire la succession des relais de messagerie à l'envers : l'en-tête contenant l'information sous la forme « reçu de la part de *w* par *x* ; reçu de la part de *x* par *y* ; reçu de la part de *y* par *z* », le chemin emprunté est, dans l'ordre chronologique, *z y x w*.

Les trois premiers relais sont des serveurs du fournisseur d'accès Free (avec les adresses IP `62.147.81.254`, `213.228.0.130` et `213.228.0.2`), les deux derniers des serveurs du fournisseur Wanadoo : *mel-rti17.wanadoo.fr* et *ms9.wanadoo.fr* (dont on n'a pas l'adresse IP publique).

Remarque

Le protocole ESMTP qui apparaît dans cet en-tête est une extension de SMTP (*Extended SMTP*). La RFC 1651 définit une commande nouvelle, EHLO (les lettres E et H inversées), qui introduit un dialogue ESMTP si les deux parties le reconnaissent. Si le serveur distant ne le reconnaît pas, la commande est ignorée et le dialogue continue en SMTP classique. Si le serveur distant reconnaît EHLO, il envoie la liste des extensions qu'il supporte et le client peut alors utiliser celles qu'il souhaite.

EXERCICE 9 MISE À DISPOSITION D'UN LOGICIEL PAR UN SERVEUR FTP

Énoncé

La société soc souhaite proposer un logiciel gratuit à télécharger sur son serveur Web (*www.soc.pays*) et place le programme correspondant *logiciel.prog* dans un répertoire FTP public `ftp/pub/freeware/`. Quelle sera l'URL pour atteindre ce produit ?

Solution

L'URL sera ftp://www.soc.pays /ftp/pub/freeware/ logiciel.prog.

Remarque

Sur les navigateurs modernes, le fait que le nom symbolique commence par www induit l'utilisation du protocole HTTP. Un serveur Web peut bien évidemment proposer des documents à transférer par d'autres protocoles que HTTP, il faut alors explicitement préciser le protocole utilisé, ici FTP.

EXERCICE 10 SERVEUR WEB SUR UN AUTRE PORT

Énoncé

Pour des raisons de sécurité, on peut être amené à développer un serveur Web sur un autre port que le port 80. Quelles seront les conséquences pour les clients ?

Solution

Le serveur Web n'est accessible que si les clients connaissent le numéro de port actif. Il faut donc publier celui-ci par exemple avec l'URL http://www.domaine.pays:8080, dans laquelle 8080 est le port utilisé.

Remarque

La RFC 1738 standardise cette écriture.

Index

T

U

V

W

X

Z

Informatique

Synthèse de cours & exercices corrigés

Architecture des réseaux

Les auteurs :

Danièle Dromard est maître de conférences à l'université Pierre et Marie Curie (Paris 6). Son domaine d'enseignement et de recherche concerne les architectures informatiques et les réseaux. Elle est responsable de l'unité d'enseignement « introduction aux réseaux » en troisième année de licence d'informatique. Elle a publié plusieurs ouvrages sur les réseaux.

Dominique Seret, professeur à l'université René Descartes (Paris 5), est directrice de l'Unité de Formation et de Recherche en mathématiques et informatique. Elle enseigne l'introduction aux réseaux en licence d'informatique ainsi que la sécurité des réseaux en master MIAGE. Son domaine de recherche porte sur les réseaux, l'évaluation de leurs performances et leur sécurité. Elle a publié plusieurs ouvrages sur ces sujets.

Cet ouvrage analyse les différents éléments qui composent un réseau, leur architecture ainsi que les protocoles de communication. Il explique tout d'abord comment les informations sont codées et envoyées sur les supports de transmission, avant de décrire les mécanismes de base d'un protocole de liaison de données. Les concepts généraux des réseaux et de leur architecture sont illustrés par de nombreux exemples. Les principaux protocoles TCP/IP, le routage et les applicatifs complètent l'exposé des caractéristiques des réseaux.

Les exercices, qui occupent la moitié du livre, sont intégralement corrigés et permettent au lecteur d'appréhender de façon progressive les différentes notions : adressage, routage, commutation, protocole et service, encapsulation, etc.

Ce livre s'adresse aux étudiants de licence, d'IUT et de BTS, ainsi qu'aux élèves ingénieurs : il se veut un cadre pratique d'apprentissage de l'architecture des réseaux informatiques ainsi qu'un précieux outil de révision. Progressif dans son approche des concepts, il constitue l'ouvrage d'initiation idéal au domaine des réseaux.

La collection *Synthex informatique* propose de (re)-découvrir les fondements théoriques et les applications pratiques des principales disciplines de science informatique. À partir d'une synthèse de cours rigoureuse et d'exercices aux corrigés détaillés, le lecteur, étudiant ou professionnel, est conduit au cœur de la discipline, et acquiert une compréhension rapide et un raisonnement solide.

Dans la même collection :

- **Le langage C**, J.-M. Léry
- **Le langage C++**, M. Vasiliu
- **Java 5**, R. Chevallier
- **UML 2**, B. Charroux, A. Osmani et Y. Thierry-Mieg
- **Création de bases de données**, N. Larrousse
- **SQL**, F. Brouard, C. Soutou
- **LateX**, J.-C. Charpentier, D. Bitouzé
- **Algorithmique, Applications en C**, J.-M. Léry
- **Algorithmique en C++**, J.-M. Léry
- **Algorithmique en Java 5**, J.-M. Léry
- **Mathématiques discrètes appliquées à l'informatique**, R. Haggarty
- **Architecture de l'ordinateur**, E. Lazard
- **Systèmes d'exploitation**, B. Lamiroy, L. Najman, H. Talbot
- **Linux**, J.-M. Léry



Pearson Education France
47 bis, rue des Vinaigriers
75010 Paris
Tél. : 01 72 74 90 00
Fax : 01 42 05 22 17
www.pearson.fr

ISBN : 978-2-7440-7385-4